Outlier Detection Using Replicator Neural Networks

Simon Hawkins, Hongxing He, Graham Williams and Rohan Baxter

CSIRO Mathematical and Information Sciences
GPO Box 664, Canberra ACT 2601, Australia
Firstname.Lastname@csiro.au

Abstract. We consider the problem of finding outliers in large multivariate databases. Outlier detection can be applied during the data cleansing process of data mining to identify problems with the data itself, and to fraud detection where groups of outliers are often of particular interest. We use replicator neural networks (RNNs) to provide a measure of the outlyingness of data records. The performance of the RNNs is assessed using a ranked score measure. The effectiveness of the RNNs for outlier detection is demonstrated on two publicly available databases.

1 Introduction

Outlier detection algorithms have application in several tasks within data mining. Data cleansing requires that aberrant data items be identified and dealt with appropriately. For example, outliers are removed or considered separately in regression modelling to improve accuracy. Detected outliers are candidates for aberrant data. In many applications outliers are more interesting than inliers. Fraud detection is a classic example where attention focuses on the outliers because these are more likely to represent cases of fraud. Fraud detection in insurance, banking and telecommunications are major application areas for data mining. Detected outliers can indicate individuals or groups of customers that have behaviour outside the range of what is considered 'normal' [8, 6, 21].

Studies from the field of statistics have typically considered outliers to be residuals or deviations from a regression or density model of the data:

> An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism [9].

In this paper we employ multi-layer perceptron neural networks with three hidden layers, and the same number of output neurons and input neurons, to model the data. These neural networks are known as replicator neural networks (RNNs). In the RNN model the input variables are *also* the output variables so that the RNN forms an implicit, compressed model of the data during training. A measure of outlyingness of individuals is then developed as the reconstruction error of individual data points. The RNN approach has linear analogues in Principal Components Analysis [10].

The insight exploited in this paper is that the trained neural network will reconstruct some small number of individuals poorly and these can be considered as outliers. We measure outlyingness by ranking data according to the magnitude of the reconstruction error. This compares to SmartSifter [22] which similarly builds models to identify outliers but scores the individuals depending on the degree to which they perturb the model.

Following [22], [4] and [17] when dealing with large databases, we consider it more meaningful to assign each datum an *outlyingness score*. The continuous score reflects the fuzzy nature of outlyingness and also allows the investigation of outliers to be automatically prioritised for analysis.

2 Related Work

We classify outlier detection methods as either *distribution-based* or *distance-based*. However, a probabilistic interpretation can often be placed on the distance-based approaches and so the two categories can overlap. Other classifications of outlier detection methods are based on whether the method provides an outlyingness score or a binary predicate (which may also be based on a score), or whether the method measures outlyingness from the *bulk* (i.e., a convex hull) of the data, or from a regression surface. Distribution-based methods include mixture models such as SmartSifter [22]. Individuals are scored according to the degree to which they perturb the currently learnt model. Distance-based methods use distance metrics such as Mahalanobis distance [2, 15] or Euclidean distance [11, 13, 12]. A prominent and useful technique for detecting outliers is to use a clustering algorithm, such as CURE or BIRCH, and then designate data occurring in very small clusters, or data distant from existing clusters as outliers [16, 21, 7, 23, 14]. Visualisation methods [3], based on grand tour projections, can also be considered distance-based since the distance between points is projected onto a 2-dimensional plane. Visualisations using immersive virtual environments [20] similarly explore the space for outliers allowing users to identify and view outliers in multiple dimensions. Despite the obvious issues of subjectiveness and scaling, visualisation techniques are very useful in outlier detection. Readily available visualisation tools such as *xgobi* [18] provide an effective, efficient, and interactive initial exploration of outliers in data (or necessarily a sample of the data). We are aware of only one other previous neural network method approach to detecting outliers [19]. Sykacek's neural network approach is to use a multi-layer perceptron (MLP) as a regression model and to then treat outliers as data with residuals outside the error bars.

3 Replicator Neural Network Outlier Detection

Although several applications in image and speech processing have used the Replicator Neural Network for its data compression capabilities [1, 10], we believe the current study is the first to propose its use as a outlier detection tool.

As mentioned in Section 1, the RNN is a variation on the usual regression model. Normally, input vectors are mapped to desired output vectors in multi-layer perceptron neural networks. For the RNN, however, the input vectors are also used as the output vectors; the RNN attempts to reproduce the input patterns in the output. During training, the weights of the RNN are adjusted to minimise the mean square error (or mean reconstruction error) for all training patterns. As a consequence, common patterns are more likely to be well reproduced by the trained RNN so that those patterns representing outliers will be less well reproduced by a trained RNN and will have a higher reconstruction error. The reconstruction error is used as the measure of outlyingness of a datum.

3.1 RNN

The RNN we use is a feed-forward multi-layer perceptron with three hidden layers sandwiched between an input layer and an output layer. The function of the RNN is to reproduce the input data pattern at the output layer with error minimised through training. Both input and output layers have n units, corresponding to the n features of the training data. The number of units in the three hidden layers are chosen experimentally to minimise the average reconstruction error across all training patterns. Heuristics for making this choice are discussed later in this section. Figure 1 shows a schematic view of the fully connected Replicator Neural Network. The output of unit i of layer k is calculated by the

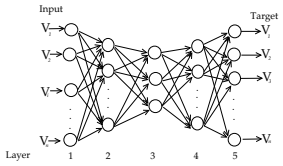

Fig. 1. A schematic view of a fully connected Replicator Neural Network.

activation function $S_k(I_{ki})$, where I_{ki}, denoted generically as θ, is the weighted sum of the inputs to the unit and defined as:

$$\theta = I_{ki} = \sum_{j=0}^{L_{k-1}} w_{kij} Z_{(k-1)j} \tag{1}$$

Z_{kj} is the output from the jth unit of the kth layer. L_k is the number of units in the kth layer.

The activation function for the two outer hidden layers ($k = 2, 4$) is then:

$$S_k(\theta) = \tanh(a_k \theta) \qquad k = 2, 4 \tag{2}$$

where a_k is a tuning parameter which is set to 1 for our experiments. For the middle hidden layer ($k = 3$), the activation function is staircase like with parameter N as the number of steps or activation levels and a_3 controlling the

transition rate from one level to the next:

$$S_3(\theta) = \frac{1}{2} + \frac{1}{2(k-1)} \sum_{j=1}^{N-1} tanh[a_3(\theta - \frac{j}{N})] \tag{3}$$

With a_3 set to a large value (we use $a_3 = 100$ throughout this work) and $N = 4$ the resulting activation function is shown in Figure 2. The activation levels of the hidden units are thus quantised into N discrete values: $0, \frac{1}{N-1}, \frac{2}{N-1}, \ldots 1$. The step-wise activation function used for the middle hidden layer divides the

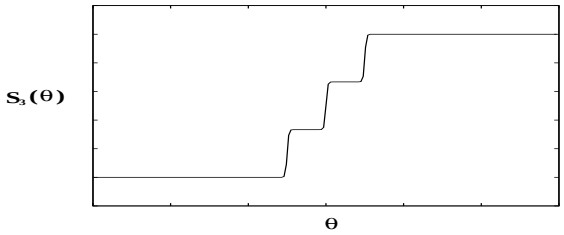

Fig. 2. Activation function of the units in the middle hidden layer.

continuously distributed data points into a number of discrete valued vectors. Through this mechanism data compression is achieved. The same architecture is adopted in this work for outlier detection. The mapping to the discrete categories in the middle hidden layer naturally places the data points into a number of clusters. The outliers identified by the RNN can be analysed further to identify individual outliers and small clusters. We discuss this further in later sections.

Training the RNN. We choose one of two candidate functions as the activation function for the output layer. The first is linear and is the weighted sum of the inputs using the formula in Equation 1, so that $S_5(\theta) = \theta$. The second is the Sigmoid function:

$$S_5(\theta) = \frac{1}{1 + e^{-a_5\theta}} \tag{4}$$

We use an adaptive learning rate for training the neural network at each iteration level, l . The weights in the neural network are updated using:

$$w_{ij}^{l+1} = w_{ij}^{l} + \alpha_{l+1} \Delta w_{ij}^{l+1} \tag{5}$$

The new learning rate at iteration $l + 1$, α_{l+1} is given by:

$$\alpha_{l+1} = \begin{cases} \beta_r * \alpha_l \text{ if } e_{l+1} > 1.01 * e_l & \text{(undo weight update)} \\ \beta_e * \alpha_l \text{ if } e_{l+1} < e_l \text{ and } \alpha_l < \alpha_{max} \\ \alpha_l \qquad \text{otherwise} \end{cases} \tag{6}$$

Where e_l in equation 6 refers to the mean square error.

$$e_l = \frac{1}{mn} \sum_{i=1}^{M} \sum_{j=1}^{n} (x_{ij} - o_{ij}^l)^2 \tag{7}$$

In Equation 7, m is the number of records in the training set, n is the number of features, and x_{ij} is the input value and is also the targeted output value ($i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$) and o_{ij}^l is the value of the output from the RNN for the lth iteration. The Initial Learning Rate, α_0, the Maximum Learning Rate, α_{max}, the Learning Rate Enlargement Factor, β_e, and the Learning Rate Reduction Factor, β_r, are adjustable parameters.

In our experiment of Section 4, we use the fixed settings shown in Table 1. . For convergence some data sets require adjustment to the training parameters

α_0	Initial Learning Rate	0.0001
α_{max}	Maximum Learning Rate	0.02
β_e	Learning Rate Enlargement Factor	1.005
β_r	Learning Rate Reduction Factor	0.98

Table 1. Parameter values used for updating the learning rate.

and the number of units in the RNN architecture. Furthermore, in the experiments to be discussed in Section 4 we use different numbers of hidden units for different data sets, ranging over 15, 35, 40, and 45. Increasing the number of hidden units increases training time but convergence is more likely. The success of training is evaluated by the average reconstruction error. It is not very sensitive to most parameters, a few experiment may be needed in choosing the values of a couple of parameters to guarantee the convergence of the error.

3.2 Methodology for applying RNN to Outlier Detection

We now describe the measure of outlyingness, the treatment of categorical variables, and the sampling scheme for large data sets.

Outlier Factor (OF) We define the *Outlier Factor* of the ith data record OF_i as our measure of outlyingness. OF_i is defined by the average reconstruction error over all features (variables)

$$OF_i = \frac{1}{n} \sum_{j=1}^{n} (x_{ij} - o_{ij})^2 \tag{8}$$

where n is the number of features. The OF is evaluated for all data records using the trained RNN.

Categorical Variables For datasets with categorical variables, we split the dataset into a number of subsets, each corresponding to a set of particular values of the categorical variables. For example, if we have two categorical variables each with two categories, we split the data set into four disjoint subsets each corresponding

to the unique combination of values of the two categorical variables. We then train an RNN for each subset individually. This is not an optimal way of treating categorical variables and a better method is being developed.

Sampling and Training For each subset C_i, we sample a portion of data either randomly or by selecting every nth record to train the RNN.

Applying the Trained RNN The trained RNN is used to calculate OF_i for all data points.

4 Experimental Results

We demonstrate the effectiveness of the RNN as an outlier detector on three data sets.

4.1 Network Intrusion Detection

We apply the RNN approach to the 1999 KDD Cup network intrusion detection data set [5]. Each event in the original data set of nearly 5 million events is labelled as an intrusion or not an intrusion. This class label is not included when training the RNN but is used to assess the RNN's performance in identifying intrusions. In summary we show that RNN outlier detection over the network intrusion data effectively identifies those events that are intrusions.

We follow the experimental technique employed in [22]. There are 41 attributes in the 1999 KDD Cup data set. There are 34 numerical variables and 7 categorical variables. The categorical variable *attack* originally had 22 distinct values (*normal*, *back*, *buffer_overflow* etc.). We map these 22 distinct values to a binary categorical variable by mapping all values, except *normal*, to *attack*. We use four of the 41 original attributes (*service*, *duration*, *src_bytes*, *dst_bytes*) because these attributes are thought to be the most important features [22]. *Service* is a categorical feature while the other three are continuous features. There are 41 original categories of *service* which are mapped into five categorical values: *http*, *smtp*, *ftp*, *ftp_data*, and *others*. Since the continuous features were concentrated around 0, we transformed each continuous feature by the log-transform $y = log(x + 1.0)$ for some subset. The original data set contained 4,898,431 data records, including 3,925,651 attacks (80.1%). This high rate is too large for attacks to be considered outliers. Therefore, following [22] we produced a subset consisting of 703,066 data records including 3,377 attacks (0.48%). The subset consists of those records with *logged_in* being positive. Attacks that successfully *logged_in* are called intrusions.

Sampling The data set is then divided into five subsets according to the five values of *service*. The aim is to identify intrusions within each of the categories by identifying outliers using the RNN approach. For the smaller of the resulting subsets (*other* contained 5858 events and *ftp* contained 4091 events) all of the

events were used to train the corresponding RNN. The subsets for *http*, *smtp* and *ftp-data* are considerably larger and were sampled in order to train the corresponding RNN within a feasible time. Note that scoring events for outlyingness is considerably more efficient than training and thus the trained RNN can be rapidly applied to very large datasets to score each event. The subsets were obtained by sampling every nth record from the data so that the subsets would be no larger than about 6K records. Details of the resultant training sets are listed in Table 2.

Service	Events	Intrusions	Proportion Intrusions	Sample	Sample Intrusions	Proportion
http	567497	2211	3.9%	5674	22	0.4%
smtp	95156	30	0.3%	5597	4	0.1%
ftp-data	30464	722	2.4%	5077	122	2.4%
other	5858	98	1.7%	5858	98	1.7%
ftp	4091	316	7.7%	4091	316	7.7%
Total	703066	3377	0.5%	26297	562	2.1%

Table 2. Summary counts of the five KDD Cup data subsets used to train each of the RNNs.

Training of the RNN The training parameters used to train the five RNNs are listed in Table 3. The choices in the training parameters were made empirically to guarantee the convergence of the mean reconstruction error to a lowest possible value for the training set.

Parameter	http	smtp	ftp-data	other	ftp
Number of activation function steps	4	4	4	4	4
Log Transform of Continuous Features	Yes	Yes	No	Yes	No
Output Layer Activation Function	Sigmoid	Sigmoid	Sigmoid	Sigmoid	Sigmoid
Number of Units: Hidden Layer 1	35	40	40	40	45
Number of Units: Hidden Layer 2	3	3	3	3	3
Number of Units: Hidden Layer 3	35	40	40	40	45
Number of Iterations	1000	40000	10000	10000	40000

Table 3. Parameter values for training the RNNs on *kddcup* network intrusion data.

Results The trained RNN is used to score each event with the value of OF. The data can then be reordered in descending order of OF. The records ranked higher are expected to be more likely intrusions. Figure 3 shows the overall ratio of the coverage of the intrusions plotted against the percentage of the data selected when ordered by OF. The combined result gives the overall performance of the outlier detector.

Best results are obtained for service type *http*, which has the highest number of intrusions. Indeed, the top one percent of the ranked records contains all intrusions. In other words, all 2211 intrusions are included in the top 5670 patterns, identifying a small data subset which has a significant proportion of outliers.

Table 4 lists the distribution of all the patterns in the codebook (identifying clusters through combination of values) of the middle hidden layer. There are

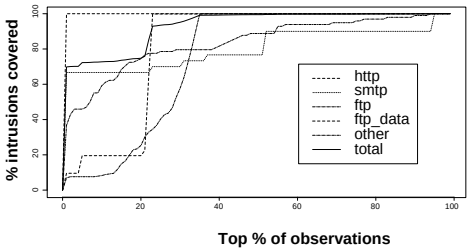

Fig. 3. Ratio of detected intrusions found by the RNN method

$N^{L_3} = 4^3 = 64$ possible codes (i.e., clusters) in the middle hidden layer. Only ten of these are non-empty clusters. It is clearly visible that cluster 48 has 2202 (99.6%) intrusion cases and only 31 normal cases. All other clusters have only normal cases (except cluster 51, which has only one intrusion cases).

Cluster Index	Middle hidden layer output Neuron 1	Neuron 2	Neuron 3	Total Records	*normal* Records	*attack* Records
0	0	0	0	21137	21131	6
1	0	0	1	103	102	1
2	0	0	2	409	408	1
16	1	0	0	539448	539448	0
17	1	0	1	50	50	0
18	1	0	2	2	2	0
32	2	0	0	4105	4105	0
48	3	0	0	2233	31	2202
49	3	0	1	9	9	0
51	3	0	3	1	0	1

Table 4. Results for *http*, according to middle hidden layer output

4.2 Wisconsin Breast Cancer Dataset

The Wisconsin breast cancer data set is found in the UCI machine learning repository[5]. The dataset contains nine continuous attributes. Each record is labelled as *benign* (458 or 65.5%) or *malignant* (241 or 34.5%). We removed some of the *malignant* records to form a very unbalanced distribution for testing the RNN outlier detection method. When one in every six *malignant* records was chosen, the resultant data set had 39 (8%) *malignant* records and 444 (92%) *benign* records. The results are shown in Table 5. Within the top 40 ranked cases (ranked according to the Outlier Factor), 30 of the malignant cases (the outliers), comprising 77% of all malignant cases, were identified. There are 25 non-empty clusters out of 64 possible codes. Among the non-empty clusters eleven of them (0-15) form a super cluster. The common characteristic of this super cluster is that the output from the first neuron is 0. There are 53 records in this cluster and 36 of them are malignant cases (outliers). We can categorise this super cluster as an outlier cluster with 68% (36/53) confidence. On the other hand, the other

three super clusters (where output from the first neuron of the middle hidden layer is 1, 2, and 3 respectively) contains mostly inliers.

Top % of record	Number of malignant	Number of record	% of malignant	Top % of record	Number of malignant	Number of record	% of malignant
0	0	0	0.00	12	35	48	89.74
1	3	4	7.69	14	36	56	92.31
2	6	8	15.38	16	36	64	92.31
4	11	16	28.21	18	38	72	97.44
6	18	24	46.15	20	38	80	97.44
8	25	32	64.10	25	38	100	97.44
10	30	40	76.92	28	39	112	100.00

Table 5. Results for Wisconsin breast cancer data according to outlier factor

5 Discussion and Conclusions

We have presented an outlier detection approach based on Replicator Neural Networks (RNN). An RNN is trained from a sampled data set to build a model that predicts the given data. We use this model to develop a score for outlyingness (called the Outlier Factor) where the trained RNN is applied to the whole data set to give a quantitative measure of the outlyingness based on the reconstruction error. Our approach takes the view of letting the data speak for itself without relying on too many assumptions. SmartSifter, for example, assumes a mixed Gaussian distribution for inliers. Distance-based outlier methods use a chosen distance metric to measure the distances between the data points with the number of clusters and the distance metric preset. The RNN approach also identifies cluster labels for each data record. The cluster label can often help to interpret the resulting outliers. For example, outliers are sometimes found to be concentrated in a single cluster (as in service type *http* of the KDD99 intrusion data) or in a group of clusters with common characteristics (as in the Wisconsin breast cancer data). The cluster label not only enables the individuals to be identified as outliers but also groups to identified as being outliers, as in [21]. We have demonstrated the method on two publicly available data sets. To test the accuracy of the method datasets with unbalanced distributions two or more classes were selected. The RNN was able to identify outliers (small classes) without using the class labels with high accuracy in both datasets. A paper comparing the RNN with other outlier detection methods is in preparation.

References

[1] D. H. Ackley, G. E. Hinton, and T. J. Sejinowski. A learning algorithm for boltzmann machines. *Cognit. Sci.*, 9:147–169, 1985.

[2] A. C. Atkinson. Fast very robust methods for the detection of multiple outliers. *Journal of the American Statistical Association*, 89:1329–1339, 1994.

[3] A. Bartkowiak and A. Szustalewicz. Detecting multivariate outliers by a grand tour. *Machine Graphics and Vision*, 6(4):487–505, 1997.
[4] M. Breunig, H. Kriegel, R. Ng, and J. Sander. Lof: Identifying density-based local outliers. In *Proc. ACM SIGMOD, Int. Conf. on Management of Data*, 2000.
[5] 1999 KDD Cup competition. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.
[6] W. DuMouchel and M. Schonlau. A fast computer intrusion detection algorithm based on hypothesis testing of command transition probabilities. In *Proc. 4th Int. Conf. on Knowledge Discovery and Data Mining*, pages 189–193, 1998.
[7] M. Ester, H. P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. KDD*, pages 226–231, 1999.
[8] T. Fawcett and F. Provost. Adaptive fraud detection. *Data Mining and Knowledge Discovery Journal*, 1(3):291–316, 1997.
[9] D. M. Hawkins. *Identification of outliers*. Chapman and Hall, London, 1980.
[10] R. Hecht-Nielsen. Replicator neural networks for universal optimal source coding. *Science*, 269(1860-1863), 1995.
[11] E. Knorr and R. Ng. A unified approach for mining outliers. In *Proc. KDD*, pages 219–222, 1997.
[12] E. Knorr and R. Ng. Algorithms for mining distance-based outliers in large datasets. In *Proc. 24th Int. Conf. Very Large Data Bases, VLDB*, pages 392–403, 24–27 1998.
[13] E. Knorr., R. Ng, and V. Tucakov. Distance-based outliers: Algorithms and applications. *VLDB Journal: Very Large Data Bases*, 8(3–4):237–253, 2000.
[14] George Kollios, Dimitrios Gunopoulos, Nick Koudas, and Stefan Berchtold. An efficient approximation scheme for data mining tasks. In *ICDE*, 2001.
[15] A. S. Kosinksi. A procedure for the detection of multivariate outliers. *Computational Statistics and Data Analysis*, 29, 1999.
[16] R. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proc. 20th VLDB*, pages 144–155, 1994.
[17] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of International Conference on Management of Data, ACM-SIGMOD*, Dallas, 2000.
[18] D. F. Swayne, D. Cook, and A. Buja. XGobi: interactive dynamic graphics in the X window system with a link to S. In *Proceedings of the ASA Section on Statistical Graphics*, pages 1–8, Alexandria, VA, 1991. American Statistical Association.
[19] P. Sykacek. Equivalent error bars for neural network classifiers trained by bayesian inference. In *Proc. ESANN*, 1997.
[20] G. Williams, I. Altas, S. Bakin, Peter Christen, Markus Hegland, Alonso Marquez, Peter Milne, Rajehndra Nagappan, and Stephen Roberts. The integrated delivery of large-scale data mining: The ACSys data mining project. In Mohammed J. Zaki and Ching-Tien Ho, editors, *Large-Scale Parallel Data Mining*, LNAI State-of-the-Art Survey, pages 24–54. Springer-Verlag, 2000.
[21] G. Williams and Z. Huang. Mining the knowledge mine: The hot spots methodology for mining large real world databases. In Abdul Sattar, editor, *Advanced Topics in Artificial Intelligence*, volume 1342 of *Lecture Notes in Artificial Intelligenvce*, pages 340–348. Springer, 1997.
[22] K. Yamanishi, J. Takeuchi, G. Williams, and P. Milne. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithm. In *Proceedings of KDD2000*, pages 320–324, 2000.
[23] T. Zhang, R. Ramakrishnan, and M. Livny. An efficient data clustering method for very large databases. In *Proc. ACM SIGMOD*, pages 103–114, 1996.