

# OPTIMUM COMPLEXITY NEURAL NETWORKS FOR ANOMALY DETECTION TASK

Robert Kozma, Nivedita Sumi Majumdar and Dipankar Dasgupta  
 Division of Computer Science, Institute for Intelligent Systems  
 373 Dunn Hall, Department of Mathematical Sciences  
 University of Memphis, TN 38152

**Abstract:** In this paper we study the performance of compressed data for classification and anomaly detection. We use networks of various complexities for our purpose, guided by the data itself rather than one uniform- complexity network for the entire data set.

## 1. INTRODUCTION

High dimensional data poses interesting challenges to computer scientists in terms of demands for a huge amount of computational resources and necessity of big training sets to ensure proper determination of weight matrices. Also, beyond a certain point, more features can actually lead to a reduction in the performance of the classification system. Data dimensionality reduction could prove very handy in this regard but arbitrary reduction in the dimension is not advisable as the loss of information could affect the classifiability of the data in a significant way. Thus there is always a tradeoff between data compression and information degradation and the selection of optimal complexity level and feature set is an important network design criteria.

Many conventional data compression techniques are available off the shelf [1]. Neural Networks have themselves been used as powerful tools for data compression [7]. In the literature there have been attempts to arrive at a single optimum compression level for the entirety of a given data set. We argue that, data sets have regions of more and less distinctiveness and therefore should dictate an optimum complexity level for each of these regions of differing distinctiveness. It is easier to detect the more distinctive regions, which may justify the use of a small number of features, a higher compression level or a smaller complexity network for their classification. Moreover, a less distinct data point would need use of more number of features, a lower compression level or a higher complexity network for their classification. A given data set could therefore be partitioned into overlapping regions according to the level of complexity required to classify members of that region.

It is an interesting observation that such a partitioning could achieve a novel characterization of the total data space. The idea is that we build networks of various complexities for the same data set. Now every point in the data set is best described by a subset of these networks. For a certain arbitrary test data point we calculate its nearest neighbor from amongst the known data members. If the test data point is normal for our data set, the same complexity networks that best classify its nearest neighbor should correctly classify it. This is because such a data point is expected to have

topographical similarity to its nearest neighbor. An incorrect classification could therefore be interpreted as an anomaly.

## 2. THE ALGORITHM

This algorithm has basically three main modules, which are, (1) Building and Tuning of the Networks; (2) Characterization of the data space; (3) Testing for Anomaly Detection. Fig. 1 describes the steps in detail

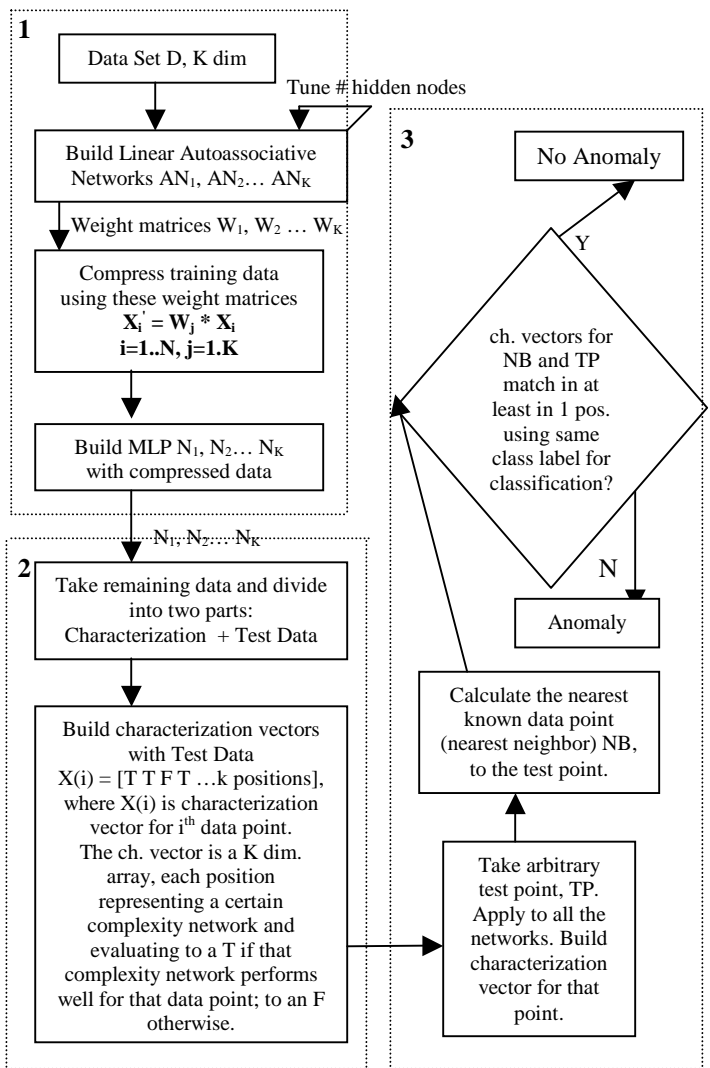


Figure 1. (b) The Algorithm in detail

## 2.1 Building and tuning of the networks

### 2.1.1 Data

The only restrictions on the format and nature of allowable data sets are that there should not be any hierarchical arrangement or hierarchical format in the data and as with all neural networks, only numeric attributes are acceptable. All non-numeric attributes needs to be preprocessed into numeric representations.

### 2.1.2 Autoassociative network layer

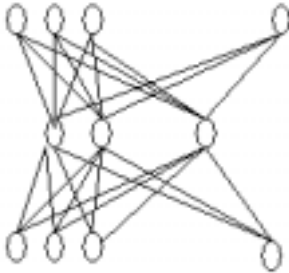


Figure 2. Autoassociative Network

Multilayer neural networks can be used to perform non-linear dimensionality reduction. We consider a multilayer perceptron having  $d$  inputs,  $d$  outputs and  $M$  hidden units, with  $M < d$  [2] & [3]. The targets used to train the network are simply the input vectors themselves, so that the network is attempting to map each input vector onto itself. Due to the reduced number of units in the first layer, a perfect reconstruction of all input vectors is not in general possible. The network can be trained by minimizing a mean of squares error of the form:

$$Performance = \sum_{j=1}^N \sum_{i=1}^M (\alpha_{ij} - t_{ij})^2, \text{ Where } N \text{ is the}$$

number of data points (in our test case  $N=178$ ),  $M$  is the number of attributes including class labels (in our test case  $M=14$ ) and  $\alpha$  is the current output of  $i^{\text{th}}$  node of the  $j^{\text{th}}$  sample point and  $t$  is the target output of  $i^{\text{th}}$  node of the  $j^{\text{th}}$  sample point.

Such a network is said to form an auto-associative mapping (fig. 2). Error minimization in this case represents a form of unsupervised training, since no independent target data is provided. If hidden units have linear activation functions, then it has been shown that the error function has a unique global minimum, and that at this minimum the network performs a projection onto the  $M$ -dimensional subspace, which is spanned by the first  $M$  principal components of the data ([4] & [5]). The limitations of linear dimensionality reduction may be overcome by using non-linear (sigmoidal) activation functions for the hidden nodes. However, it was shown by Boulard and Kamp [4] that such non-linearities make no real difference.

We use Data Set  $D$  of  $K$  dimensions to build autoassociative Neural Networks of varying complexities. The number of hidden nodes is representative of this complexity level. We denote the weight matrices thus obtained as  $W_i$ , where  $i$  is the number of hidden nodes. This achieves compression of the  $K$  dimensional data into  $i$  dimensions. This represents stage 1 of figure 3.

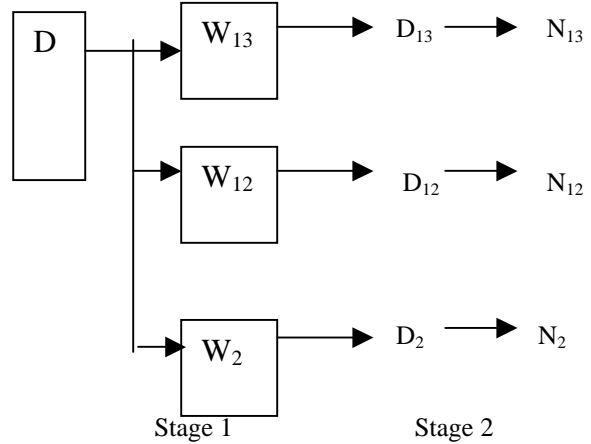


Figure 3. Schematic Diagram of the steps

### 2.1.3 Compress train data set

The train data is compressed to various degrees using the compression weight matrices already derived from our previous phase as follows:

$X_i' = W_j * X_i$  Where  $X_i$  is a sample train data point and  $X_i'$  is that data point compressed to  $j$  dimensions.

Thus we generate:

$$D_{13} = W_{13} * D$$

$$D_{12} = W_{12} * D$$

.

.

And so on.

### 2.1.4 Architecture of the MLPs

We build MLPs  $N_1, N_2, \dots, N_K$  with the compressed data sets  $D_1, D_2, \dots, D_K$  respectively. (Stage 2 of Figure 3). We fine-tune the number of hidden nodes in the second layer of the MLP. Table 1 shows the values derived in successive runs experimenting with different number of nodes for each complexity network. The optima are shown in bold face for reference.

Then we apply each of the data points of the test data set to all our networks  $N_1, N_2, \dots, N_K$  and obtain misclassification ratios for each of these networks  $MSR_1, MSR_2, \dots, MSR_K$ . During the evaluations, we graph the misclassification ratio values against the level of compression, as well as the performance values against the level of compression to see how compressed data is performing the task of classification.

## 2.2 Characterization of the data space according to the optimal complexity networks

We split the test data for characterization and testing. For each test data point we build its characterization vector as follows: The characterization vector is a  $K$  dimensional vector, each position  $1...K$  representing a different network complexity level. For each data point in the test set, if network  $i$  performs well enough for it, we put a T in its characterization vector position  $i$ , an F otherwise. Therefore a vector [T T F F T T F F F T F F F] would mean that networks 1,2,5,6,10 perform well for it. We hope that the members of the test data set make a good representation of the entire data space so that we can use this representation for anomaly detection.

The idea is as follows. Figure 4 shows a schematic diagram of the entire data space covered by overlapping circles. Each circle represents a certain network complexity level and the points in the data space it classifies correctly. The circles are overlapping because the same point may be well classified by more than 1 network. Also intuitively, topologically close data points will share similarity and therefore should be easy to detect with the same networks and therefore such circles are justifiable.

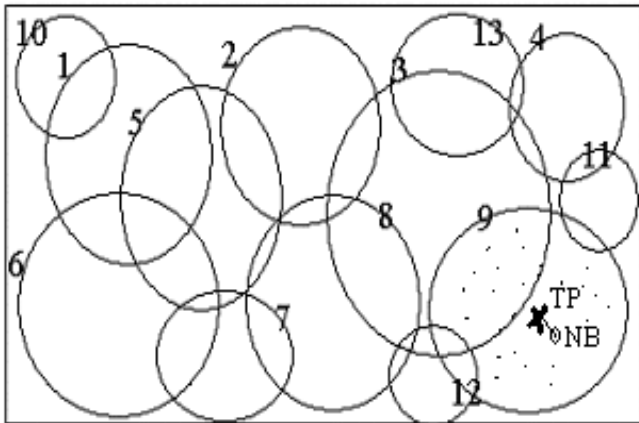


Figure 4. Conceptual diagram illustrating the characterization of the data space with NNs of different complexities, numbered from 1 to 13.

## 2.3 Anomaly detection methodology

The data points, which rightfully belong in the data set, are “normal” points for it. However, due to noise, data corruption, transmission errors and various other reasons, certain data points may get introduced in the data set, which are not quite “normal” in the context of the data. To devise a mechanism for the system to identify such mistakes is the essence of anomaly detection [8]. There are many standard techniques for anomaly detection. [9] & [10]. Figure 4 illustrates the logic used by our system for detection of anomalies. An arbitrary point TP is taken and applied to all the networks. Its characterization vector is built as outlined before. Then we find its nearest neighbor, which is the point marked with a circle in Figure 4. It is our assumption that,

Network 9, which describes NB well, should also describe TP well. Therefore we compare the characterization vectors of NB and TP. If we can identify a point  $i$  and  $j$  in the characterization vectors of NB and TP respectively, such that position  $i$  has a T value for characterization vector of NB and position  $j$  has a T value for characterization vector of TP and moreover, the class label output by network  $i$  for NB matches the class label output by network  $j$  for TP, then we can call point TP as a normal point. If there is no match or conflicting class labels are generated, then we consider TP as an anomalous point. It is expected that since the test data set we use at the moment is part of the data set really, so all points there should be classified as normal. Our experimental results reported in the next section confirm this expectation. For experimenting with negative data samples, we trained the system with samples from only one class and used the other class samples as anomalous examples. Here, an anomalous classification for all those points is achieved as is reported in the next section.

## 3. IMPLEMENTATION

### 3.1 Description of the Analyzed Data

We used the normalized wine database for our experiments obtained from the UCI Machine Learning Data Repository [6]. These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

The attributes are (donated by Riccardo Leardi, riclea@anchem.unige.it) 1) Alcohol 2) Malic acid 3) Ash 4) Alkalinity of ash 5) Magnesium 6) Total phenols 7) Flavanoids 8) Nonflavanoid phenols 9) Proanthocyanins 10) Color intensity 11) Hue 12) OD280/OD315 of diluted wines 13) Proline

There are a total of 178 instances used in the experiments, of which 59 belonged to class 1, 71 belonged to class 2 and 48 belonged to class 3.

### 3.2 Experimental results

Experimentation results from various stages of the algorithm are shown in the following sections of the paper. Table 1 shows the number of hidden nodes exhaustively experimented with for a given compression level in regard to the MLPs. Finally the number of hidden nodes with the minimum number of misclassification for each compression level is chosen for the actual testing.

Figure 5 displays the misclassification ratio against network complexity. Figure 6 shows the Mean Squared Error Performance Curve versus number of hidden nodes in Autoassociative Networks. As it is expected, the Performance gets worse as the bottleneck layer gets narrower. When we use all 13 attributes, in the hidden layer, the error is clearly close to zero. The weight matrix at that point showed that the

corresponding node position weight connections had a value close to 1, where as the other weight connections had values close to zero. The plateau like nature of the curve when the number of hidden nodes is between 6 and 8 show that those values give equally good performance, so using 7 nodes instead of 8 will not alter the performance.

The curve is better interpreted if we compare it to the Misclassification Ratio Curve versus number of hidden nodes in Autoassociative Networks given in figure 5. Clearly compression below 6 hidden nodes is not advisable as performance as well as misclassification drastically increases in that case.

Samples of the characterization vector for the test data is given in Table 2. When we experiment with our testing data set, which is part of the whole data set, we get very few anomalies reported. This is close to our expectation of no detection at all, as the data being tested with is really part of the whole data set. When we experiment with an arbitrary point, clearly anomaly is reported by the system. We present the readings in Table 3.

To exhaustively experiment and test the limitations of the technique, we gradually introduce some percentage of error to our dataset and test the method with it. The following error function is used:

$E = \alpha * (r - 0.5)$ , where  $r$  is a random number between 0 and 1 and  $\alpha$  is a tunable parameter. The graph in Figure 7 plots the number of anomalies reported with increasing value of  $\alpha$ . The system does not begin to misclassify badly until the value of  $\alpha$  goes to 3. The reason for this behavior is that one of the classes is very easily separated from the two others; even with the compression level as high as from fourteen to two dimensions. Therefore the structure is not easily destroyed.

We try to construct the anomaly detection rates and report the results in Table 4. The positive samples detected as non-anomalies or anomalies as the case may be are easy to obtain and the results are reported for that. In 78.9% cases we see there is no detection as it should be and for 21.1% cases there is false detection signals. The challenge was really to test it with negative data, as that is difficult to devise. But we trained the system with class 1 samples and tested with the remaining data using exactly same process as before. This time we obtained a 100% anomaly detection and no false positives.

		Number of hidden Nodes																		
		2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Network Complexity	13	7	7	5	5	3	1	1	3	2	1	2	2	2	2	2	2	3	3	3
	12	7	7	7	8	8	3	2	3	2	4	3	3	2	3	2	2	2	5	5
	11	10	9	10	10	8	6	2	7	5	4	2	4	5	5	3	3	3	4	4
	10	11	6	6	6	5	6	5	2	6	8	6	6	4	3	5	5	3	4	2
	9	7	6	5	4	3	4	3	4	3	3	1	2	2	2	3	3	3	4	7
	8	8	7	7	6	5	5	5	3	2	3	1	2	3	4	5	4	4	5	4
	7	15	16	15	15	9	6	5	7	15	8	10	12	14	16	15	16	15	13	14
	6	10	9	7	7	5	3	2	6	7	8	8	7	6	8	7	5	6	7	6
	5	23	24	21	21	21	18	14	22	24	19	20	18	19	29	20	19	18	20	15
	4	18	15	14	12	13	10	10	14	8	12	12	13	11	13	13	15	16	10	16
	3	17	17	15	19	17	14	16	19	15	17	17	15	16	17	17	15	13	16	
	2	18	16	15	16	15	13	16	15	18	14	15	14	15	12	10	12	16	15	12

TABLE 1. NETWORK COMPLEXITY VERSUS HIDDEN NODES

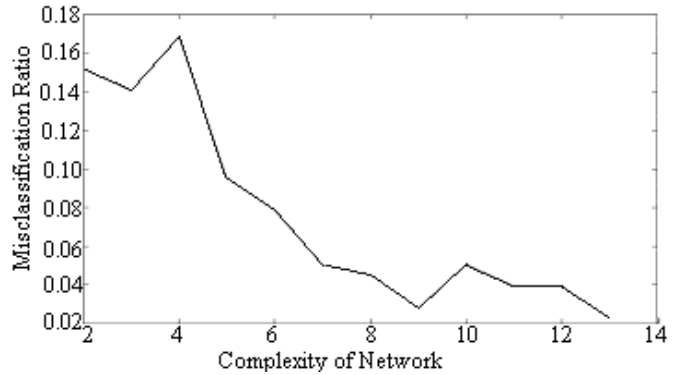


Figure 5. Misclassification Ratio Curve versus number of hidden nodes in Autoassociative Networks

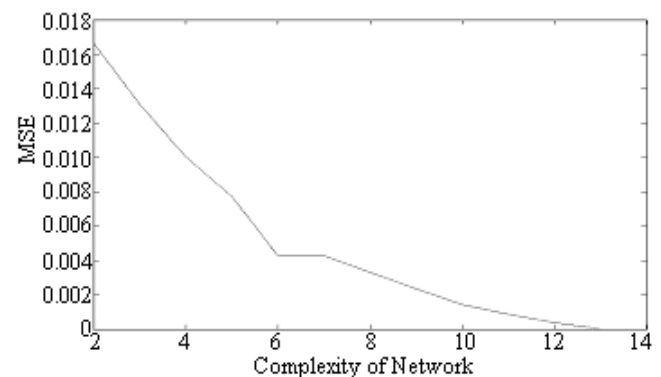


Figure 6. Mean Squared Error Performance Curve versus number of hidden nodes in Autoassociative Networks

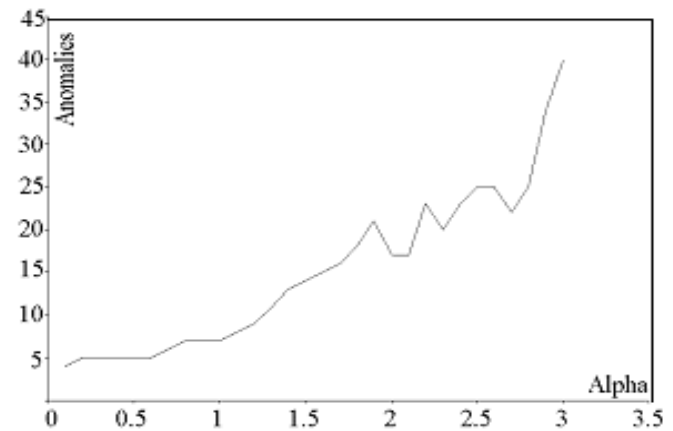


Figure 7. Number of Anomalies versus alpha

Characterization_Vector =
TTTFTFFFF
...
...
...
TTFFTTFFTFF
TTTFTTFFFF
TTTFTTFFFF
TTTFTTFTFF
TTTFTTFFFF

TABLE 2. CHARACTERIZATION VECTOR FOR TEST DATA SET

<p>T = [0.2 0.4 0.5 0.1 0.4 0.5 0.2 0.9 0.1 0.2 0.8 0.1]</p> <p>NB: T T T F T T F F F F F F F F</p> <p>TP: F F F F F F F F F F F F F F</p> <p>Reports ANOMALY</p>
---

TABLE 3. SAMPLE TEST RESULTS

Detection Rate	NO ANOMALY (Averaged over 4 runs)	ANOMALY (Averaged over 4 runs)
POSITIVE SAMPLES (45 tier)	78.9%	21.1%
NEGATIVE SAMPLES (119 tier)	0%	100%

TABLE 4. ANOMALY DETECTION TABLE

4. CONCLUSIONS

In this study we see that appropriately compressed data not only performs well for classification purposes but also can be used for anomaly detection applications with remarkably good performance. This is quite an unexpected, somewhat surprising result. It can be interpreted as follows. The loss of information in an information theoretical sense after data compression may not always mean a loss in practical sense. Namely it may imply a gain in terms of extra knowledge acquired in identifying the more important components and the basic structure of the data. This notion is illustrated in the present study. Data-driven system characterization is one of the most fundamental themes of Computational Intelligence and it is the key to the success of our approach. Class 1 samples of wine data used here are fairly easily separable from the other two class samples even when they are compressed to two dimensions. Therefore the negative samples were easier to detect. In the future, this method will be further tested with complex data sets in order to test the applicability of our method in a wide range of anomaly detection problems.

ACKNOWLEDGEMENTS

This work has been supported in part by NSF grant number NSF-IIS-0104251. We would also like to thank Jonathan Gomez and Fabio Gonzalez for their help and useful comments.

5. REFERENCES

- [1]. Salomon David. Data Compression: The Complete Reference. 2<sup>nd</sup> edition (October 2000) Springer Verlag.
- [2]. Rummelhart D., and McClelland, J., (eds.), 1986., *Parallel Distributed Processing: Explorations into the Microstructure of Cognition, Volume 1: Foundations*, The MIT Press, Cambridge, Mass.
- [3]. Rumelhart D., G.E. Hinton and R. J. Williams 1986., *Learning internal representations by error propagation*.
- [4]. Boulard, H. and Y. Kamp (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics* 59, 291-294.
- [5]. Baldi, P. and K. Hornik (1989). Neural networks and principal component analysis: learning from examples without local minima. *Neural Networks* 2, 53-58.
- [6]. UCI Machine Learning Data Repository. <http://www1.ics.uci.edu/~mllearn/MLRepository.html>
- [7]. Bishop, C. Pattern Recognition using Neural Networks.1995, Oxford University Press.
- [8]. Liepins G. E. and H. S. Vaccaro. Anomaly Detection: Purpose and Framework Proc. 12th NIST-NCSC National Computer Security Conference, pp. 495-504, 1989.
- [9]. Dipankar Dasgupta and Stephanie Forrest, 'An Anomaly Detection Algorithm Inspired by the Immune System', Chapter 14 in the book entitled Artificial Immune Systems and Their Applications, Publisher: Springer-Verlag, Inc., pp 262-277, January 1999.
- [10]. Dipankar Dasgupta, Using Immunological Principles in Anomaly Detection, In the proceedings of the Artificial Neural Networks in Engineering (ANNIE'96) Conference, St. Louis, November 10-13, 1996.
- [11]. Lane T., and C. E. Brodley. An Application of Machine Learning to Anomaly Detection. Proc. 20th NIST-NCSC National Information Systems Security Conference, pp. 366-380, 1997.