

Use of Instance Typicality for Efficient Detection of Outliers with Neural Network Classifiers

Shirish S. Sane
Department of Computer Engineering
Pune Institute of Engineering & Technology
Pune, Maharashtra, India.
sssane@sify.com

Ashok A. Ghatol
Vice-Chancellor
Dr. B. A. Technological University
Lonere, Maharashtra, India.
vc_2005@rediffmail.com

Abstract

Detection of outliers is one of the data pre-processing tasks. In all the applications, outliers need to be detected to enhance the accuracy of the classifiers. Several different techniques, such as statistical, distance-based and deviation-based outlier detection exist to detect outliers. Many of these techniques use filter method. A wrapper method using the concept of instance typicality may also be used to detect outliers. This paper deals with a new wrapper method that builds an initial model using neural networks and treats values at the output of neurons in the output layer as the typicality scores. Instances with lowest output values are treated as potential outliers. In addition, the method is also useful to build compact and accurate classifiers by selecting a few most typical instances resulting in significant reduction in storage space. The method is generic and thus can also be used for instance selection with any kind of classifiers. Resultant compact models are useful for imputation of missing values.

Keywords: Classification, Data Mining, Instance-Typicality, outliers.

1. Introduction

Data mining is a process of finding useful patterns in data. It is successfully used in large diversified applications such as credit approvals, detecting fraudulent activities in telecommunications, classifying patients for presence or absence of a disease, prediction of rain falls, market basket analysis for sales promotions, prediction of trends in stock market etc. and are discussed in [1, 2, 3, 4] and elsewhere. Data mining deals with several different types of tasks such as classification, prediction, clustering, association mining, sequence co-relation etc. Classification is one of the commonly used tasks to solve many real-life problems such as credit approvals. Classification is a supervised learning method. In supervised learning, a model is trained using a set of training instances

consisting of known values for input and output attributes. If the information regarding the values of output attribute(s) for the given set of input attributes, is not available, the instances are clustered into two or more clusters, based on the similarities of the instances, using what is called unsupervised clustering technique. Each of the clusters is treated as a class and assigned a unique class label. All instances in a cluster have class label of that cluster. Using the set of training instances, the model is trained and used to classify unseen instances into one of the pre-determined classes. It is observed that, the outliers, if not detected and eliminated, may significantly affect the accuracy of a classifier. Data mining algorithms therefore need to detect and eliminate outliers or at least minimize the effect of outliers. Several different techniques exist to detect outliers [6, 7, 8]. Instance typicality as discussed in [5] can also be used to efficiently detect candidate outliers.

This paper deals with how the concept of instance typicality can be used to detect outliers and build compact and accurate classification models. Section 2 deals with outliers, role of instance typicality is discussed in section 3, experimental results are provided in section 4 and section 5 concludes the paper.

2. Outliers

Very often, in large datasets, there exist samples that do not comply with the general behavior of the data model. Such samples that are significantly different with the remaining set of data are called outliers. Outliers can be caused by measurement error or they may be the result of inherent data variability.

Many data mining algorithms try to minimize the influence of outliers on the final model, or to eliminate them in the data pre-processing phase. However, a data miner should be careful while automatically detecting and eliminating outliers because, if the data are correct, that could result in the loss of important hidden information [4]. Some data mining applications are focused on outlier detection, and it is the essential result of a data-analysis. For

example, while detecting fraudulent credit card transactions, the outliers are typical examples that may indicate fraudulent activity, and the entire data mining process is concentrated on their detection. But, in most of the other applications of data mining, typically with large data sets, outliers are not very useful.

Several different techniques exist to detect outliers such as statistical method, distance and deviation based methods etc. Statistical method is simple but makes apriori assumption about data distribution and in most of the real-world applications, data distribution is unknown. Deviation based method do not make any assumptions about the distribution of the data and is applicable to multi-dimensional samples. Deviation based method defines the basic characteristics of the sample set and all samples that deviate from these characteristics are treated as outliers. The method can be very complex and the problem of selection of set of potential outliers is theoretically defined as NP-hard problem.

Concept of Instance typicality as discussed in [5] may also be used for efficient detection of the outliers using the instance-based classifiers. This paper illustrates how a neural network classifier may also implement this concept.

3. Instance Typicality

Instance typicality of an instance I , defined in [5, 9] as, the average similarity of I to the other members of its class. Highly typical data instances within a class are called class prototypes and instances with low typicality scores represent candidate outliers. It is observed that by selecting representative training data from a pool of available instances and limiting the inclusion of atypical instances, test set classification accuracy can be improved and also results into compact supervised classification models. Experimental results presented in section 4 confirm this finding.

3.1 Using Instance Typicality With Neural Network Classifiers

A multi-layer feed forward neural network classifier has one input layer, one or more hidden layers and one output layer. Number of neurons in input layer depends on number of input attributes selected by the variable selection algorithm and number of transforms generated for each of the selected input attributes. Number of neurons in the output layer depends on number of target class labels. Generally, a neural network classifier uses one neuron in the output layer for problems with two classes. If number of target classes is greater than two, then ' n ' neurons are used in the output layer one for each of the target classes. For the two class problems having classes, say class 0 and class 1, assuming a

decision boundary at 0.5, the neuron in the output layer generates the outputs between 0.0 and 0.5 for all the instances that are classified into, say, class 0. For the instances that are classified into class 1, the neuron output is in the range of 0.5 to 1.0. The actual output of the neuron is transformed back to real-world values to indicate the actual class labels. For multi-class problems with n target classes, the classifier normally designates *one* neuron in the output layer for each of the n target classes. For example, a neural network classifier may use three output neurons to represent three target classes (Class 1, 2 and 3) in IRIS flower dataset [11]. Classification decision is provided in the form of 1-of- N code. Thus, instances that belong to class 1, the output neuron corresponding to class 1 will have values between 1.0 and 0.5 and the other two neurons will have values between 0.0 to 0.5. Instances that belong to class 2, the designated output neuron for class 2 will have values between 1.0 and 0.5 and the other two neurons will have values between 0.0 to 0.5 and so on.

In case of a two-class problem, if the values at the outputs of the neurons in the output layer are observed, it can easily be understood that not all instances belonging to class 0 have values close to 0.0 and not all instances belonging to class 1 have values close to 1.0. Consider two instances I_1 and I_2 from class 0 such that the output values for the two instances are say, 0.0023 and 0.4728 respectively. When the model converts these actual values into real-world values, the classifier will assign class labels 0 to both the instances. However, instance I_1 is classified into class 0 by the network with higher confidence than the confidence used while classifying instance I_2 into class 0. Jimenez in [10] defined this as certainty of the network and used it for building dynamically weighted neural network ensembles. If instance typicality is computed, typicality score for instance I_1 should be higher than that of typicality score for instance I_2 . Therefore, the actual output values of the neuron in the output layer should be treated equivalent to the typicality scores and can be used to identify prototype instances, outliers and also to build compact but accurate classification models.

3.2 Use of Instance Typicality for the Detection of Outliers and to build compact models

Method to detect the outliers using the concept of instance typicality is as given below.

1. Using the available training set, train the model using neural network classifier
2. Classify all instances from training set using the trained model and record the output values of the neurons in the output layer for each of the training instances.

3. Sort the records from training set on class labels and then sort records in each of the classes using the values at the output of a neuron in the output layer designated for the class. Now, most typical instances for each class will appear first and least typical instances will appear last. The least typical instances for each class are indeed candidate outliers.
4. Select most typical records by choosing equal number of instances having output values at the output of the neuron in the output layer in the range $1.0 - 0.9$, $0.9 - 0.8$, $0.8 - 0.7$, $0.7 - 0.6$ and $0.6 - 0.5$.

4. Experimental Results

This section presents results of a few experiments made by the authors. The objectives of performing these experiments were

1. To verify similarity between instance typicality scores and values at the outputs of the neurons in the output layer.
2. To validate the method for the detection of outliers discussed in section 3.2
3. To construct compact and accurate classification models.

Publicly available datasets such as BUPA, Credit and Housing [11] were used to perform experiments.

All experiments were performed using trial version of NeuralWare Predict, a product developed by NeuralWare downloadable from <http://www.neuralware.com> that provides neural network classifier. Algorithm presented in section 3.2 was implemented in C. The experiments were performed using default setting of parameters only.

For each dataset, classification models were constructed using the standard technique of 10-fold cross-validation. Algorithm presented in section 3.2 was executed to determine most typical and least typical instances. Models were constructed using most typical and least typical instances. Table 4.1 shows the details of the average accuracy, size of the disk file in kilobytes (that stores the resultant network along with other auxiliary information) and the complexity of the resultant model in terms of number of nodes in input, hidden and output layers. For example, the average accuracy of the classifier for the dataset Bupa, trained using 10-fold cross validation with all available instances is 67.1%. Average storage space required is 27.3 KB and on an average, there are eight neurons in the input layer, 3 neurons in the hidden layer and two neurons in the output layer. Algorithm presented in section 3.2 determines the most and least typical instances. The average accuracy of the classifiers for the dataset Bupa, constructed using only the most

typical instances is 67.43% and it is slightly higher than the models constructed using all the instances. The average size of the disk file is 4.6 KB as compared to 27.3 KB and thus resulting in significant reduction in storage space. The average complexity of the model is 6-2-2, i.e. six neurons in the input, two each in hidden and output layers as compared to 8-3-2, the complexity of models constructed using all available instances. Models constructed using the most typical instances with all the datasets shown in Table 3.1 results into significant reduction in storage space compared to the models constructed using all the instances. Average number of neurons in the hidden layer is also lesser than average number of neurons required for models constructed using all the instances. However, in case of Credit and Housing datasets, the average classification accuracy of the models with unseen instances constructed using the most typical instances is slightly lesser than that of the accuracy of the models constructed using all available instances. This is because; only 20% of the available input instances were used to construct the models. In order to verify that the values at the outputs of the neurons in the output layer indicate the typicality of the instances, models were constructed using the least typical instances for each datasets. Table 4.1 shows the performance of these models. Use of least typical instances in the training set results into significant reduction in the classification accuracy. For example, the average classification accuracy of the models for the dataset Bupa using the least typical instances is only 38.53 as compared to 67.1%. For the credit dataset, the average accuracy is 34.41% as compared to 86.83% and average accuracy in case of Housing dataset is 78.59% as compared to 91.53%. Table 4.2 provides details of the datasets such as number of instances used for training, testing, class distribution and number of most and least typical instances used for training. As mentioned earlier, for the datasets with two classes, the models may be constructed either using one neuron in the output layer or with two neurons. With one neuron in the output layer, the outputs for correctly classified instances belonging to one of the classes is in the range of 0.0 to 0.5 and 0.5 to 1.0 for the correctly classified instances belonging to the other class (assuming the decision boundary at 0.5). With two neurons in the output layer, all correctly classified instances have values in the range of 0.5 – 1.0 for both the classes. Models for datasets Bupa and Housing were designed with two neurons in the output layer while there was only one neuron in the output layer for models constructed for the Credit dataset. The code written in C handles any of these cases. With only one neuron in the output layer, the decision boundary may not be always at 0.5 and the code also can handle such situations. Authors have conducted similar experiments with several other datasets such as IRIS flower, PIMA, Wisconsin Breast-

Cancer, Cardiology, Wine, US-House-Vote-84 and Banning datasets etc. to test the algorithm and obtained similar results with all of these datasets. The method is generic and thus can be used with any kind of classifiers. For example, an instance-based or decision

tree classifier may use the neural classifier to build initial model, run the algorithm to select the instances and build a compact or a tiny model. Neural network classifiers may use the algorithm to build tiny models that may be used for imputation of missing values.

Table 4.1 Performance of Neural Network Classifier Using values at the outputs of the neurons in the output layer

Dataset	Classification Model Using								
	All Instances			Most Typical Instances			Least Typical Instances		
	Accuracy	Size	Comp	Accuracy	Size	Comp	Accuracy	Size	Comp
Bupa	67.1	27.3	8-3-2	67.43	4.6	6-2-2	38.53	8.9	9-3-2
Credit	86.83	104	7-3-1	85.14	18.6	8-1-1	34.41	21.4	17-2-1
Housing	91.53	64.6	7-4-2	90.49	10	9-1-2	78.59	11	12-2-2

Table 4.2 Details of datasets

Dataset	Average No. of Training Instances	Average No. of Unseen Instances	Class Distribution	No. of Most Typical Instances	No. of Least Typical Instances
Bupa	315	30	42% - 58%	50	115
Credit	500	190	54% - 46%	105	105
Housing	450	50	80% - 20%	90	90

5. Conclusions

Many data mining systems detect and eliminate or at least minimize the influence of outliers. Several different techniques such as statistical method, distance and deviation-based techniques are commonly used for the detection of outliers. Concept of instance typicality generally used by the instance-based classifiers can efficiently be used not only to detect outliers but also select a few typical instances to build compact yet accurate classification model. This paper presents a novel wrapper method to detect potential outliers using the values at the outputs of the neurons in the output layer. Experiments discussed in this paper discover the similarities between instance typicality and values at the output of the neurons in the output layer of a neural network. These values that are readily available in the network have been used in ensemble techniques. This paper demonstrates how these values may also be used as instance typicality scores to detect outliers and also to build compact and accurate classification models. Section 3 illustrates how outliers can be detected and compact yet accurate models can be constructed. It results into significant reduction in storage space without affecting the classification accuracy. Existing neural network classifiers can easily incorporate the concept of instance typicality to detect outliers and construct compact but accurate models. The wrapper approach presented in this paper is a general-purpose instance selection method and thus can be used with any kind of classifiers such as instance-based or decision tree classifiers. It is also useful to build tiny neural networks for the imputation of missing values.

References

- [1] Berry, M., Linoff, G. *Mastering Data Mining The Art and Science of Customer Relationship Management*, 2nd Ed., Wiley Publishers, 2003.
- [2] Delmater, R., Hancock, M. *Data Mining Explained: A Manager's Guide to Customer-Centric Business Intelligence*, Digital Press, 2001.
- [3] Han, J., Kamber, M. *Data Mining Concepts and Techniques*, Morgan Kaufmann Publishers, 2001.
- [4] Kantardzic, M. *Data Mining Concepts, Models, Methods, and Algorithms*, Wiley – Interscience Publications, IEEE Press, 2003.
- [5] Roiger, R., Geatz, M. *Data Mining A Tutorial-based Primer*, Pearson Education, 2003.
- [6] Angiulli, F., Pizzuti, C., "Outlier Mining in Large High-Dimensional Data Sets", *IEEE Trans. KDE*, vol. 17, No. 2, pp. 203 – 215, Feb. 2005.
- [7] Angiulli, F., Basta, S., Pizzuti, C., "Distance-based detection and prediction of outliers", *IEEE Trans. KDE*, vol. 18, no. 2, pp. 145 – 160, Feb. 2006
- [8] Takeuchi, J., Yamanishi, K., "A unifying framework for detecting outliers and change points from time series, *IEEE Trans. KDE*, vol. 18, no. 4, pp. 482- 492, 2006.
- [9] Morring, B., Martinez, T., "Weighted instance typicality search (WITS): A Nearest Neighbor data reduction algorithm", *Intelligent Data Analysis*, 7(6), 2003.
- [10] Jimenez, D., "Dynamically weighted ensemble neural networks for classification", In: *Proc. of IEEE Int'l joint Conf. on Neural Networks*, Anchorage, AK, vol. 1, pp. 753 – 756, 1998.
- [11] Blake, C.L., and Merz, C. J. *UCI Repository of Machine Learning Databases*, <http://www.ics.uci.edu/~mllearn/MLRepository.html>.