Graph Regularized Autoencoder and its Application in Unsupervised Anomaly Detection

Welcome to the online companion (Code and Datasets) of our paper. There are two subfolders located under the root folder along with this README file. The contents of these two subfolders are explained below:

Datasets:

This subfolder stores all the datasets required for generating the results in the paper. We used 20 benchmark datasets for the performance evaluation. Several versions of these data sets are stored in the online repository of [1] (https://www.dbs.ifi.lmu.de/research/outlier-evaluation). These versions mainly differ in terms of the preprocessing steps used and the proportion of anomalies compared to the normal observations. Table 3 in the paper summarizes the basic characteristics of these 20 data sets used in our study. All of these 20 datasets are stored in .csv format. The last columns of these .csv files indicate the anomaly information (1 indicates an anomaly and 0 indicates a normal observation). To reproduce the anomaly detection results for two datasets ('KDD' and 'ALOI'), it is recommended to use cloud service or high performance research computing (HPRC) environment. However, for the demonstration purpose, we provide a smaller version of both of these datasets (kdd_short and aloi_short). For the other 18 datasets, personal computing resources are enough to reproduce the results reported in the paper.

Codes:

This subfolder contains all the scripts and functions required to generate results in the paper. The scripting language used includes Python (primary), R and Matlab. In Python, we used several libraries that need to be installed and imported during the execution of IPython codes. The required libraries are listed below:

- Tensorflow 1.5.0
- SciPy 1.5.4
- Numpy
- Pandas
- Scikit-learn

Similarly in R, we used several packages apart from the basic packages and they are required to be installed and loaded before running any of the R code (.r) files. The required packages are listed below:

- dbscan
- dplyr
- fossil
- intrinsicDimension
- RANN
- PCCMR
- matrixStats
- vegan3d
- ider

Under the Codes folder, we have several IPython notebooks, a couple of R and Matlab scripts & functions. The main code files are explained below:

'datasetname'MDSLE.ipynb: These IPython notebooks will generate the F1-score following the MST regularized autoencoder model using both MDS and LE formulation. There are 20 such notebooks corresponding to each of the 20 datasets.

'datasetname'euclidMDSLE.ipynb: These IPython notebooks will generate the F1-score following the Euclidean distance regularized autoencoder model using both MDS and LE formulation. There are 20 such notebooks corresponding to each of the 20 datasets.

'datasetname'GAE.ipynb: These IPython notebooks will generate the F1-score following the GAE model [2]. There are 20 such notebooks corresponding to each of the 20 datasets.

'datasetname'noreg.ipynb: These IPython notebooks will generate the F1-score following the original autoencoder model without any regularizer. There are 20 such notebooks corresponding to each of the 20 datasets.

'datasetname'Denoise.ipynb: These IPython notebooks will generate the F1-score following the regular and special denoising option. There are 20 such notebooks corresponding to each of the 20 datasets.

ID.r: This R script will generate the intrinsic hidden layer dimension for all of the 20 datasets.

AnoGAN-MST-paper.ipynb: It will generate the F1-score using with and without regularizer version of the AnoGAN approach for the KDDCup dataset.

ALAD-MST-paper.ipynb: It will generate the F1-score using with and without regularizer version of the ALAD approach for the KDDCup dataset.

LoMSTcof.r: It will generate the F1-score using the autoencoder generated hidden layer results following LoMST and COF approach.

friedman.r: It will first perform the Friedman test on the detection results generated by competing approaches. Then it will generate the p-value table reported in Table 6 in the paper.

GNMF-Cluster.m: It will generate GNMF's anomaly detection performance on MNIST and COIL-20 dataset reported in Table 11 and 12.

MNISTGAE.m and COALGAE.m: It will generate GAE's [2] anomaly detection performance on MNIST and COIL-20 dataset reported in Table 11 and 12. These code files are directly collected from the first author of the GAE [2] paper.

MNISTMST.m and COAL-MST.ipynb: It will generate our MST regularizer's anomaly detection performance on MNIST and COIL-20 dataset reported in Table 11 and 12.

Figure3.r: It will recreate the Figure 3 of the paper.

Figure4.m: It will recreate the Figure 4 of the paper.

Figure5.ipynb: It will recreate the Figure 5 of the paper.

Figure6&7.ipynb: It will recreate the Figure 6 and 7 of the paper.

Table9.ipynb: It reproduces the Table 9 reported in the paper

Apart from these main scripts we have several other helper functions that are required for compiling the above scripts or generate data files used in those scripts, they are briefly mentioned below:

LoMST.r: It is the core LoMST algorithm function. The function will be called to return the number of true detection and anomaly indices for a specified K value for any dataset whose anomaly information is known to the user.

ClusterTest.m: It will run the K-means clustering algorithm on the data to generate two clustering performance indices (Normalized Mutual Informaiton and Clustering Accuracy).

GNMF.m: It will run the GNMF algorithm. The code was taken from GNMF author's website.
bigankdd_utilities.py: It is a utility function to compile the AnoGAN approach
bn.py: It is a utility function to compile the ALAD approach
kdd.py: It loads and preprocess the KDDcup dataset to be used for the GAN approaches
dagmm.py: It is the main core function of the DAGMM approach

Machine Specification: Intel Core i7(7700HQ@2.80 GHz), 16GB Ram; Windows 10
Software Version: Python 3.6; R version 3.6.2; MATLAB version 2019b

Reproducing the results in the paper:

For the convenience of the users, in the following table, we summarize how to reproduce different tables and figures used in the paper. Before running any codes, set your working directory to ".\Codes".

Which Results to Reproduce	Data File	Code File	Output
Figure 3	Swissroll.csv Swirleuclid.csv SwirlMST.csv	Figure3.r	Figure 3(a), 3(b), 3(c) and 3(d)
Table 3	All 20 datasets (.csv)	ID.r	Column 5 of Table 3 (Intrinsic dimension of 20 datasets)
Figure 4	Paper34.csv	Figure4.m	Figure 4 (Post-hoc analysis)
Table 5	All 20 datasets (.csv)	'datasetname'MDSLE. ipynb* (Column 4, 5) 'datasetname'euclidMDSLE.ipynb* (Column 2, 3) 'datasetname'GAE.ipynb* (Column 6) 'datasetname'noreg.ipynb* (Column 7) ***Replace 'datasetname' with the respective data file name**	Table 5 (F1-score returned by individual approaches for all 20 datasets); Table 5 is used to manually generate Table 4
Table 6	Paper314.csv	friedman.r	Table 6 (P-value from friedman test)
Table 7	'datasetname'm.csv* ***Replace 'datasetname' with the respective data file name**	LoMSTCOF.r	Column 2 and 3 of Table 7

Table 8	All 20 datasets (.csv); All 20 datasets with noise ('datasetname'denoise.csv)	'datasetname'Denoise. ipynb* ***Replace 'datasetname' with the respective data file name**	Table 8 (F1-score following regular and special denoising option for all 20 datasets)
Figure 5	WPBC.csv; WDBC.csv; Cardioto.csv	Figure5.ipynb	Figure 5(a), 5(b), 5(c) and 5(d)
Figure 6, 7	WBC.csv; WDBC.csv; Arrh.csv; Spambase.csv; Cardioto.csv; WPBC.csv; Ionosphere.csv; Pblock.csv; Lymph2.csv	Figure6&7.ipynb	Figure 6(a), 6(b), 6(c), 6(d) and 7
Table 9	Kdd.csv; Aloi.csv; Arrh.csv	Table9.ipynb	Table 9 (F1-score using state of art approaches)
Table 10	Kddcup	AnoGAN-MST-paper.ipynb (Row 2-5) ALAD-MST-paper.ipynb (Row 6-9)	Table 10 (GAN approaches with and without MST regularizer)
Table 11	subMNIST.mat	MNISTMST.m (Row 2) MNISTGAE.m (Row 3) GNMFCluster.m (Row 4)	Row 2,3,4 of Table 11 (Results of row 5,6,7 are directly taken from [2])
Table 12	COIL20.mat	COIL-MST.ipynb (Row 2) COILGAE.m (Row 3) GNMFCluster.m (Row 4)	Row 2,3,4 of Table 12 (Results of row 5,6,7 are directly taken from [2])

[1] G. O. Campos et al., "On the evaluation of unsupervised outlier detection: Measures, datasets, and an empirical study," Data Mining and Knowledge Discovery, vol. 30, no. 4, pp. 891–927, 2016.

[2] Y. Liao et al., "Graph regularized auto-encoders for image representation," IEEE Transactions on Image Processing, vol. 26, no. 6, pp. 2839–2852, 2016.

****Thank you for using this online companion. If you have any questions on implementing our algorithm, please feel free to send an email at** imtiaz_avi@yahoo.com