Autoencoder-based One-class Classification

Conference Paper · November 2019

CITATIONS
0

READS
148

2 authors:

Isuru Jayarathne
The University of Aizu
6 PUBLICATIONS **68** CITATIONS
SEE PROFILE

Michael Cohen
The University of Aizu
186 PUBLICATIONS **1,131** CITATIONS
SEE PROFILE

計測自動制御学会東北支部 第 326 回研究集会 (2019.11.15)
資料番号 326-7

Autoencoder-based One-class Classification

○ イスル ジャヤラトナ *，公園 マイクル *

○Isuru Jayarathne* and Michael Cohen*

* 会津大学コンピュータ理工学部

*University of Aizu

Keywords: Autoencoder，One-class classification，Neural Networks

〒 965-8580 福島県会津若松市 会津大学コンピュータ理工学部
Tel.: (0242)37-2615，Fax.: (0242)37-2747，E-mail: {d8191101, mcohen}@u-aizu.ac.jp

Abstract

One-class classification (OCC) technique based on autoencoders is proposed in this study. OCC can be especially used in the user authentication domain when filtering imposter data. Six different autoencoder configurations were tested for performances using the publicly available MNIST dataset. 94.51% maximum average accuracy of all classes was achieved for a five-layer, 128 code layer nodes configuration. All the configurations showed 0.96 average AUC value. According to the results, the proposed approach can be used as a one-class classifier with high accuracy in the image processing domain.

1. Introduction

Binary or multi-class classification tries to find a boundary (or boundaries) between classes used in various domains, including, but not limited to, image, signal, and text processing. Such classification techniques usually output the closest class label for any given input. Therefore, binary classification techniques cannot be used without some modification, especially in user authentication applications. However, most biometric authentication systems use pattern matching [1] or distance measurement [2] techniques, which do not necessarily use multiple classes in the training phase. One-class classification (OCC), a.k.a. Unary classification, which was coined by Moya and Hush in 1996 [3], can also be used to solve this problem. OCC is a method of identifying objects of a specific class amongst all objects, by learning from a training set that comprises only objects of that class. While traditional binary classification uses data of two classes, OCC techniques learn from single class. The most popular OCC algorithms are Support Vector Data Description (SVDD) [4], which finds the smallest hypersphere that contains all objects of a specific class, and one-class SVM (OC-SVM) [5], which separates inliers from outliers by finding a hyperplane of maximal distance from the origin.

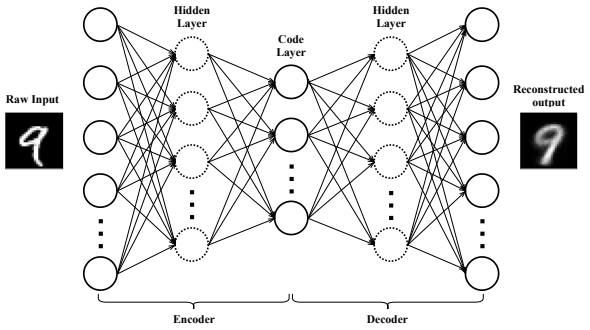


Fig. 1: General structure of autoencoder

An autoencoder is a type of ANN which comes under the dimensionality reduction (or compression) category in ML (machine learning). It basically learns a representation (encoding) of a set of data in an unsupervised manner. This concept was first proposed by Geoffrey Hinton and the PDP group in the 1980s to address the problem of "back-propagation without a teacher" [6]. In a training phase, an autoencoder extracts some important features in the encoding phase from raw data to make an abstract representation. Reconstructed output can be obtained as output of the decoder. This technique is effectively used for denoising, visualizing high dimensional data, and anomaly detection. The basic structure of an autoencoder comprises three layers, but additional hidden layers can also be added, as shown in Figure 1, and both encoder and decoder are fully-connected feedforward neural networks [7]. The middle layer, called the "bottleneck" or code layer, holds compressed data, and the network is usually symmetric across this code layer. However, several variants with different numbers of layers can be used depending on the application. The number of nodes in input and output layers must be the same to reconstruct same-sized output. Besides using fully-connected layers, there are some autoencoder models with convolution layers, called deep autoencoders [8].

2. Methodology

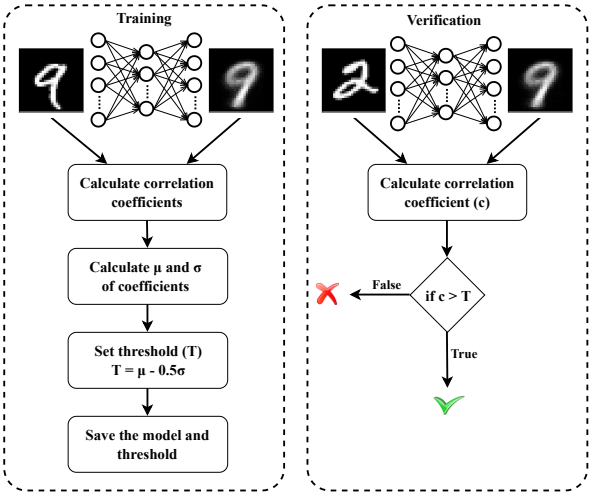
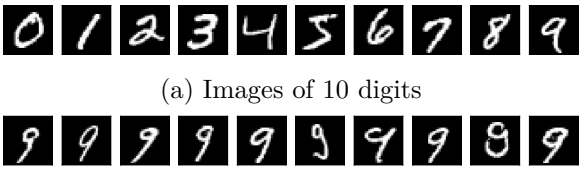


Fig. 2: Structure of training and verification phases

As outlined in the previous section, the reconstructive nature of autoencoders is used to implement new OCC algorithms. Typical autoencoders follow the same structure as ANNs, so each node has connections with all nodes in the next layer (fully-connected layer), as in Figure 1. In our case, the ReLU (rectified linear units) activation function is used in every layer. In training, as the decoder reconstructs encoded data, it determines the error compared to actual data and adjusts the weights to minimize the error. The binary cross-entropy loss function is used because it is usually applied to problems involving binary decisions. To update the weights of the network, the Adam (Adaptive Momentum) [9] optimizer is used. As seen in Figure 2, first, the autoencoder is trained with one specific class. Then, correlation coefficients are calculated with actual (x) data and the predicted data (y).



(a) Images of 10 digits

(b) Sample (variant) images of digit 9

Fig. 3: Sample images of MNIST dataset

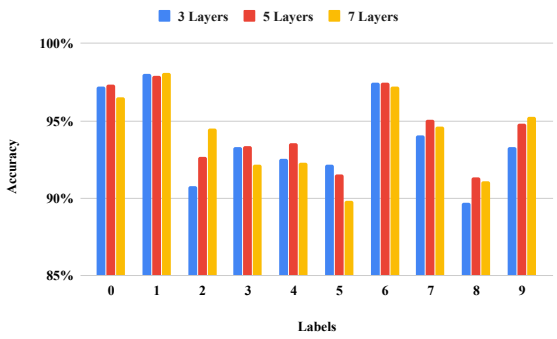
$$Corr(x,y) = \frac{\Sigma_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\Sigma_{i=1}^{n}(x_i - \overline{x})^2 \Sigma_{i=1}^{n}(y_i - \overline{y})^2}}, \quad (1)$$

where x_i, y_i are sample values and $\overline{x}$, $\overline{y}$ are means of two populations (pixel grayscale values of images). A threshold T is calculated using mean (μ) and standard deviation (σ) of an array of correlation coefficients:

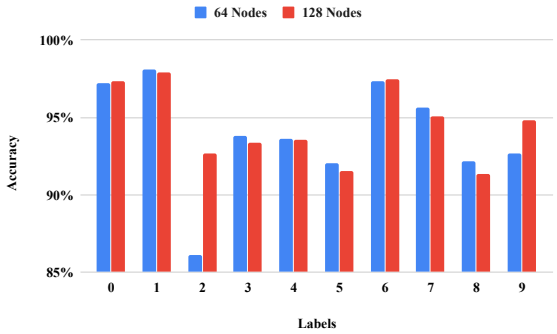
$$T = \mu_c - 0.5\sigma_c, \quad (2)$$

where μ_c is the mean of that class, σ_c is the standard deviation, and scaling coefficient 0.5 was determined heuristically (by trial and error), After saving the trained model and the threshold, the trained model can be used as an OCC for the trained class. In the verification phase, the correlation coefficient is calculated using the given data and corresponding prediction. The calculated value is compared with the threshold, and if greater, the given data is considered in the same class.

Before testing with EEG data, a simple autoencoder model was designed for a publicly available dataset, the MNIST dataset, which contains 70,000 images [10]. This dataset includes classified images corresponding to handwritten digits 0–9 with resolution 28×28 pixels. It has been split into two sets — 60,000 training and 10,000 testing images — to reduce the effort of pre-processing. Figure 3a shows sample images of 10 handwritten digits, and Figure 3b shows different sample images



(a) Comparison of layer architecture with 128 nodes code layer



(b) Comparison of code layer nodes of 5 layers configuration

Fig. 4: Comparison of accuracies

of digit 9. Normalized pixel values of these grayscale images are between 0 and 1.

Implementation was done in Python using the Tensorflow [11] and Keras [12] libraries. Three variants of autoencoders with three, five, and seven fully-connected layers were used to test performance of the proposed method. The three-layer architecture was extended by adding a pair of layers each with 256 nodes to create five layers. Another pair of layers each with 512 nodes was added to the five-layer architecture to make seven layers. Input and output layers always contain 784 (28×28) nodes, which is the total number of pixels in the image. Two code layers with 64 and 128 nodes were also tested with all three autoencoder configurations.

3. Results and Discussion

There is no substantial difference among accuracies in autoencoder configurations as seen in Figure 4. Five- and seven-layer configurations, showed slightly better accuracy compared to the basic three-layer configuration, as seen in Figure 4a. Moreover, the code layer with 128 nodes showed better accuracy in node configuration comparison, as seen in Figure 4b. According to the results, 94.51% average accuracy of all classes was achieved for the five layers and code layer with 128 nodes configuration.

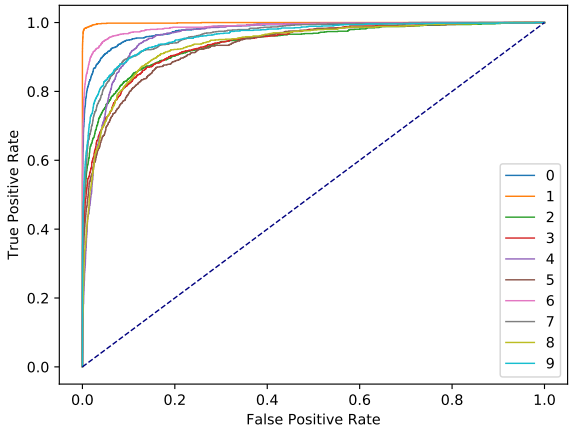


Fig. 5: ROC curve of three layers and code layer with 64 nodes

Receiver operating curve (ROC) and area under the curve (AUC) values were calculated with the calculated correlation coefficient values for each testing sample. As seen in Figure 5, the threshold can be selected with a high true positive rate (TPR) and a low false-positive rate (FPR) for most of the classes. TPR, a.k.a. sensitivity or recall, measures the proportion of actual positives that are correctly classified and FPR, a.k.a. 1− specificity, measures the proportion of actual negatives wrongly identified. If the TPR value is 1 while FPR 0 (AUC also 1), such a classifier can be considered perfect [13]. Recognition power of digit 1 is maximum over other classes, and digit 8 showed lowest performance. Table 1 shows calculated AUC values for all configurations, and all values are over 0.9. The average AUC value of all classes and all the configuration is 0.960 ± 002. Therefore, even the smallest tested configuration can be used for classification.

4. Conclusion

According to results, the proposed approach can successfully be used as a one-class classifier. When using this method, each class can be trained independently, which is an advantage for dynamic class problems such as user authentication. Traditional classification algorithms need to be retrained when adding each new class, and the accuracy of the model can be highly affected. Our approach can be used for signature recognition which involves image recognition.

References

[1] M. Aljamea, T. Athar, C. S. Iliopoulos, S. P. Pissis, and M. S. Rahman, "A novel pattern matching approach for fingerprint-based authentication," in *PATTERNS: Seventh Int. Conf. on Pervasive Patterns and Applications*, 2015, pp. 45–49.

[2] F. Sufi, I. Khalil, and I. Habib, "Polynomial distance measurement for ecg based biometric authentication," *Security and Communication Networks*, vol. 3, no. 4, pp. 303–319, 2010.

[3] M. M. Moya and D. R. Hush, "Network constraints and multi-objective optimiza-

Table 1: AUC values of all configurations

Configuration	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
3 layers, 64 nodes	0.981	0.999	0.939	0.937	0.960	0.930	0.987	0.962	0.937	0.962
5 layers, 64 nodes	0.989	0.999	0.941	0.954	0.967	0.927	0.984	0.932	0.920	0.968
7 layers, 64 nodes	0.988	0.999	0.944	0.909	0.944	0.936	0.981	0.966	0.940	0.968
3 layers, 128 nodes	0.992	0.999	0.949	0.955	0.966	0.937	0.985	0.964	0.911	0.964
5 layers, 128 nodes	0.983	0.999	0.928	0.952	0.954	0.932	0.984	0.968	0.926	0.960
7 layers, 128 nodes	0.989	0.999	0.949	0.962	0.952	0.933	0.981	0.965	0.925	0.962

tion for one-class classification," *Neural Networks*, vol. 9, no. 3, pp. 463–474, 1996.

[4] D. M. Tax and R. P. Duin, "Support vector data description," *Machine Learning*, vol. 54, no. 1, pp. 45–66, 2004.

[5] L. M. Manevitz and M. Yousef, "One-class svms for document classification," *J. of Machine Learning Research*, vol. 2, no. Dec, pp. 139–154, 2001.

[6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," UCSD La Jolla Inst. for Cognitive Science, Tech. Rep., 1985.

[7] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proc. ICML Workshop on Unsupervised and Transfer Learning*, 2012, pp. 37–49.

[8] K. G. Lore, A. Akintayo, and S. Sarkar, "Llnet: A deep autoencoder approach to natural low-light image enhancement," *Pattern Recognition*, vol. 61, pp. 650–662, 2017.

[9] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[10] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[11] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symp. on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[12] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[13] T. Fawcett, "An introduction to roc analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.