Proceedings of the 2017 International Conference on Wavelet Analysis and Pattern Recognition, Ningbo, China, 9-12 July, 2017

IMPROVING ROBUSTNESS OF STACKED AUTO-ENCODER AGAINST EVASION ATTACK BASED ON WEIGHT EVENNESS

FENGZHI LUO[1], PATRICK P.K. CHAN[1],ZHE LIN[1], ZHIMIN HE[2]*

[1]School of Computer Science and Technology, South China University of Technology, Guangzhou 51006, China
[2]School of Electronic and Information Engineering, Foshan University, Foshan 528000, China
E-MAIL: lllffz@163.com , patrickchan.scut@gmail.com, chirplam@foxmail.com, zhmihe@gmail.com (corresponding author)

Abstract:

In recent years, deep learning makes great breakthroughs in a variety of applications. Its motivation is to learn feature from raw data. However, the performance of deep learning method applied to security-sensitive applications is poor. In adversarial environment, there are adversaries who attempt to evade valid classification by manipulating training and test samples. Thus, test data does not follow the same distribution of training data. Robustness of stacked autoencoder must be concerned under adversarial environment. In this paper, we investigate the attacking principle and try to improve the robustness of stacked autoencoder. A simple but effective objective function that not only minimizes empirical risks but also improves the robustness of stacked autoencoder against evasion attack is proposed. Our proposed method is based on weight evenness which presents better classification performance under evasion attack. We verify this method on the MNIST dataset. The robustness of our model is compared with conventional stacked autoencoder. The experimental results show that the proposed method is effective against evasion attack and enhances the security of stacked autoencoder under adversarial environment.

Keywords:

Evasion Attack; Weight Evenness; Robustness

1. Introduction

Deep learning is a mighty machine learning technique. Different from traditional machine learning methods such as SVM [1], deep learning learns feature representation hierarchically and automatically [2]. With such methodology, handcraft features are no longer needed. Since learned representation can express the data in an efficient manner. Deep learning methods obtain excellent performances on many applications, including image classification [3-5], object detection [6-8], speech recognition [9, 10] and natural language processing [11, 12].

A representative deep learning method is stacked autoencoder [13]. Stacked autoencoder learns hierarchical features from raw data (e.g. image). There are two training phases of stacked autoencoder, i.e., pre-training and fine-tuning. Pre-training is unsupervised training phase which aims to extract features from input. The learned features should retain the most important information of its input. The learned features should be able to reconstruct the input with minimum loss. Fine-tuning is supervised learning phase which tunes the learned features of pre-training. A supervised fine-tuning procedure aims to fit the features for classification task.

However, deep learning methods are not always applied to stationary environment. Some applications are security-sensitive, including biometric recognition [14, 15], spam filtering [16, 17] and instruction detection [18]. In adversarial environment, there exists an adversary who manipulates the data carefully at test time, in order to make the modified samples successfully escape the detection of a classifier. The success of deep learning on adversarial environment is related to its ability of resistance to adversarial samples. Current researches reveal that deep learning methods are not robust to abnormal inputs [19]. Small perturbation on input can easily cause misclassification [20, 21]. The performance of deep learning significantly drops in adversarial environment. Hence, the robustness of deep learning methods on adversarial environment should be concerned. Stacked autoencoder as one of deep learning method, its robustness against evasion attack is not well studied.

In this paper, we firstly explore the vulnerability of stacked autoencoder facing evasion attack. A robust learning algorithm balances the classification capacity and the robustness to evasion attack. Our main idea is to train an even weight stacked autoencoder. Compared with conventional stacked autoencoder, our proposed model will consider more kinds of features. As a result, more modification must be done on input samples and the cost of evasion attack increases. The rest of paper is organized as follows. Section 2 introduces

978-1-5386-0412-0/17/$31.00 ©2017 IEEE

stacked autoencoder and adversarial environment. Section 3 exhibits our proposed method. Section 4 compares our method with conventional stacked autoencoder. The conclusion is given in Section 5.

2. Improving robustness based on weight evenness

2.1. Stacked autoencoder

Model capacity of neural network is highly depended on the depth of network (i.e. number of hidden layers). Nevertheless, neural network with more hidden layers gets worse result compared with shallow network [2]. Deep neural network using back propagation algorithm to train will have instable gradient. Updates on weight will be too tiny or too large, which makes training difficult. In order to handle such problem, pre-training is used in stacked autoencoder. Pre-training is a greedy layer-wise unsupervised learning. Representative features are learned with unsupervised training. Then, back propagation algorithm is applied to the whole network to further adjust the weights learned in pre-training stage.

Each time the weight is trained so that it can extract main features from input. Let the input of current hidden layer be $\mathbf{h}$. Feature extracted from input is denoted as $\mathbf{y}$,

$$y = Wh + b, \tag{1}$$

where $\mathbf{W}$ is an weight matrix and $\mathbf{b}$ is bias. Then we reconstruct the original input from $\mathbf{y}$ by

$$z = W'y + b \ , \tag{2}$$

where $\mathbf{W}'$ is an weight matrix and $\mathbf{b}'$ is the bias. The weight and bias are learned so that the reconstruction error of $\mathbf{x}$ and $\mathbf{z}$ is minimized. The most common measure of error is the mean square error:

$$J_{pre} = (h - z)^2 / 2 . \tag{3}$$

After parameters in hidden layers are trained, fine-tuning is applied to adjust the connection weights. Fine-tuning is supervised learning and the label is used. We denote the input of network is $\mathbf{x}$ and the corresponding label is y. The network gives a prediction of $\mathbf{x}$ which is $\mathrm{g}(\mathbf{x})$. Fine-tuning aims to minimum the prediction error which is the different between y and $\mathrm{g}(\mathbf{x})$. The mean square error is used here as well:

$$J_{fine\text{-}tuning} = \frac{(g(x) - y)^2}{2} . \tag{4}$$

Generally, the prediction error of the whole training set is taken into account during fine-tuning stage.

2.2. Adversarial attack

Machine learning usually assumes that the distribution of test data is the same as training data. In adversarial environment, the assumption does not hold due to the existence of adversary [29, 30]. The adversary attempts to evade detection of classifier by modifying input [20, 21] while the defender proposes methods [23] to reduce the attack influence and enhance security.

In this paper, the worst scenario is considered in this paper. The adversary is assumed to have perfect knowledge of model (i.e., discriminate function of classifier). Adversary will try to minimize the modification of input. We denote the modified input as $\mathbf{x}'$. There is different distance measure between $\mathbf{x}$ and $\mathbf{x}'$. Two kinds of attacks (sparse attack and dense attack) [23, 24] are defined according to different distance measure. Dense attack changes all features in the input. The Euclidean distance between $\mathbf{x}$ and $\mathbf{x}'$ is minimized. Sparse attack minimizes the number of modified features. The attack cost is related to the number of modified features. In this paper, we focus on sparse attack. The modification of input is considered as an optimization problem:

$$\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}'} L_0(\mathbf{x}, \mathbf{x}') \tag{5}$$

$$\text{s.t. } f(\mathbf{x}') \neq f(\mathbf{x}) \tag{6}$$

where f is the classification function. L_0 counts the number of different features between $\mathbf{x}$ and $\mathbf{x}'$. Its formula is given as follows

$$L_0(\mathbf{x}, \mathbf{x}') = \sum_{i=0}^{m} (x_i - x'_i)^0, \tag{7}$$

where x_i and x'_i is the ith element in $\mathbf{x}$ and $\mathbf{x}'$ respectively. We assume there are m elements in $\mathbf{x}$ and $\mathbf{x}'$ respectively. Such optimization can be solved by Algorithm 1.

As we can see, each time the gradient of output probability of class y with respect to $\mathbf{x}'$ is computed. And the index of smallest element of gradient is return. Modifying the corresponding features in $\mathbf{x}'$ will decrease the probability of $\mathbf{x}'$ belongs to class y. Each modification on feature is to set the feature to maximum value. For example, pixels in an image can be set to 255. A threshold is set to limit the number of features to be modified. If the number of modified features exceed the limitation, the attack is treated as failure.

Algorithm 1: **Sparse Attack**

Input: $\mathbf{x}$: the original input; f: the classification function; g_y: a function output the probability of a given input belongs to class y; $d_{\max}$: the maximum number of features allows to be modified; U: the maximum value of a feature.
Initialize: $\mathbf{v}$: an vector contains m zeros; τ is set to 0
Output: $\mathbf{x}'$: adversarial sample which is misclassified.

```
x' ← x
while f(x') = y and τ < d_max :
    index = argmin_i ∇g_y(x') and v_i ≠ 1
    v_index ← 1
    x'_index ← U
    τ ← τ+1
return x'
```

2.3. Weight evenness

Feature reweighting [25, 26] is proposed to improve the robustness of classifier. When a classifier is learned, important features is expected to have higher weights. The adversary will use such properties to evade the detection of classifier by only modifying features with high weights. Feature reweighting first learn to be a classifier and then redistribute the weight in order to make it more even. The redistribution is implemented by modifying the training data according to the learned features and retraining the classifier. This training method is not suitable for stacked autoencoder since training time of stacked autoeoncoder is very long.

A feature weights with even distribution forces an adversary to modify more features and increase the cost of evasion attack [27]. Feature reweighting first learn a classifier and then redistribute the weight in order to make it more even. The redistribution is implemented by modifying the training data according to the learned features and retrain the classifier. Such training method is not suitable for stacked autoencoder since training time of stacked autoencoder is very long. The formula of weight evenness defined in [24] is given as follow:

$$E = \frac{2}{d-1}\left[d - \sum_{k=1}^{d}\left(\sum_{i=1}^{k}|w^{(i)}| / \sum_{j=1}^{d}|w^{(j)}|\right)\right]$$
$$\text{s.t. } |w^{(1)}| \geq |w^{(2)}| \geq \cdots \geq |w^{(d)}| \quad , \quad (8)$$

where $|w^{(i)}|$ is the absolute value of ith weight in a classifier. Totally there are d weights on the classifier. Weight evenness in Equation (8) is a good measurement of weight evenness. However, it requires sorting operation of weight, which is not convenient to apply to the training of stacked autoencoder directly. In the following section, we use a simple criterion to implement weight evenness.

3. Proposed method

Training a stacked autoencoder means selecting one model from the set of allowed models that minimizes the cost criterion essentially. Most of the learning algorithms used in training artificial neural networks employ some form of gradient descent, using back propagation to compute the actual gradients [22]. This is done by simply taking the derivative of the cost function with respect to the network parameters and then changing those parameters in a gradient-related direction. Traditional training strategy only minimizes training error regardless of robustness. Consequently, trained stacked autoendoer will be vulnerable to sparse attack. An adversary can change a few features to cause misclassification.

We consider the vulnerability of conventional stacked autoencoder coming from its non-uniform rely on a few set of features. In other words, stacked autoencoder makes classification on several features and the rest of features are regardless. Hence, an adversary only needs to modify a few important features to cause misclassification with little cost.

The importance of features depends on their weights. In order to avoid the loss caused by sparse attack, the weight of each feature of the trained classifier should be distributed evenly and we should reduce the possibility that the classifier is too dependent on certain characteristics. Therefore, the weight distribution of each feature of the trained classifier is expected to be relatively uniform. To overcome the shortcoming of distribution of features being uneven, we introduce an additional component into our training objective function, which is given as follow.

$$\boldsymbol{J_{fine-tune}} = \frac{(\mathrm{g(x)-y})^2}{2} + \lambda \sum_{k=0}^{n}\sqrt{\sum(\mathbf{W_{ij}}^{\boldsymbol{k}} - \bar{\mathbf{W}}^{\boldsymbol{k}})} \qquad (9)$$

where $\sum_{k=0}^{n}\sqrt{\sum(\mathbf{W_{ij}}^{\boldsymbol{k}} - \bar{\mathbf{W}}^{\boldsymbol{k}})}$ is the standard deviation of weight in k-th hidden layer. $\mathbf{W_{ij}}^{\boldsymbol{k}}$ is element in the i-th row and j-column of weight matrix while $\bar{\mathbf{W}}^{\boldsymbol{k}}$ is the average of all elements in weight matrix of k-th layers. λ is the tradeoff parameter which controls the importance of robustness. From Equation (9), the weight evenness is represented by standard deviation. The smaller the standard deviation is, the more even the weight is. Even weights in hidden layers make features have similar importance. Stacked autoencoder not only classifies input based on some importance feature, but takes many other features into consideration. Thus, the robustness of stacked autoencoder against evasion attack is improved.

One may notice that we only modify the cost function of fine-tuning stage instead of pre-training. That is because minimizing standard deviate of weight during pre-training stage makes little effort. Without weight evenness in fine-

tuning stage, the even weight learned in pre-training stage will be replaced by uneven weight in order to minimizing the classification error. Thus, we choose to adjust uneven weight learned by pre-training during fine-tuning stage, which is more effective.

4. Experiment

In this section, we evaluate our proposed model under evasion attack on MNIST dataset. Weight evenness is applied to different training phase, including pre-training and fine-tuning. Effectiveness of weight evenness against evasion attack is compared with traditional stacked autoencoder.

4.1. Experiment setting and criterion

In the experiment, misclassification error and average modified features (i.e., pixels in image) is tested under sparse attack. While performing evasion attack, the adversary always focusses on cutting down perturbation on test input. MNIST dataset is used for evaluation.

Weight evenness is considered in unsupervised, supervised of stacked autoencoder separately. We compare our method with stacked autoencoder and demonstrate effectiveness of sensitivity. Generally, three models are trained: stacked autoencoder (SAE), stacked autoencoder with pretraining weight evenness (SAE-pre) and stacked autoencoder with fine-tuning (SAE-fine) weight evenness. All these three models contain an input layer, three hidden layers and one output layer. The hidden layer size is 900. Softmax regression is used in output layer for classification. Specially, the tradeoff parameter λ is set to 10 and 20 respectively.

MNIST dataset is an image dataset containing 70,000 samples. Images on MNIST dataset are gray-scale. The range of pixel is in [0, 254]. In order to make training easier to convergence, we compress it to [0, 1]. The MNIST dataset is spilt to training set and test set. Training set contains 56000 samples while test set contains 14000 samples. We implement the algorithm using Theano framework [28]. Five independent runs are performed to evaluate the generalization error and robustness of three models.

4.2. Result and discussion

In Table 1, we give out the classification accuracy of different models. The average classification accuracy of SAE achieves 97.61%. When trade off parameter lambda is set to 10, the classification accuracy of SAE-pre does not drop significantly. Similar situation happens on SAE-fine. When lambda is set to 20, the accuracy of SAE-fine drops to 93.54%. However, the classification accuracy of SAE-fine remains the same.

TABLE 1. Classification accuracy of models under different settings

Model	Lambda	classification accuracy
SAE	/	97.61%
SAE-pre	10	97.60%
SAE-fine		96.13%
SAE-pre	20	97.61%
SAE-fine		93.54%

In Table 2, we present the classification accuracy under sparse attack. The number of maximum features allowed to be modified is set to 4, 8 and 16. With the increase of modified features, classification accuracies of three models drop. SAE gives the worst performance. When the number of maximum modified features is set to 16, SAE only gets 2.89% accuracy on test data. Meanwhile, compared with SAE, SAE-pre does not preserve much classification accuracy. The ability of SAE-fine against sparse attack performs much better than SAE and SAE-pre. When the number of maximum modified feature is 4, SAE-fine still gets 79% accuracy which is about 20% higher than accuracies of SAE and SAE-pre.

TABLE 2. Classification accuracy of models under different sparse attack level

model	lambda	classification accuracy		
		4	8	16
SAE	/	59.94%	22.00%	2.89%
SAE-pre	10	59.95%	21.93%	2.88%
SAE-fine		**77.40%**	**49.62%**	**11.97%**
SAE-pre	20	59.93%	22.15%	2.96%
SAE-fine		**79.04%**	**56.76%**	**16.95%**

We present the number of features needed in order to cause misclassification. In this experiment, all test samples are misclassified with sufficient modification. The average modified features of five independent runs is given in Table 3. From Table 3 we can see numbers of average modified features of SAE and SAE-pre are similar. Pre-training with weight evenness does not improve the robustness of stacked autoencoder a lot. The number of modified features of SAE-fine is 1.6 times as large as SAE. The attack cost of adversary is significantly increased by our proposed method. Besides, we increase the tradeoff parameter lambda and observe its influence. However, the number of modified features does not increase a lot with the increase of lambda. Setting lambda to 10 is enough to increase the robustness of stacked autoencoder in MNIST dataset. Large value of lambda cannot further improve robustness but do harm to classification accuracy.

TABLE 3. Number of modified features to successfully evade detection

model	Lambda	modified features
SAE	/	7.14
SAE-pre	10	7.14
SAE-fine		**10.48**
SAE-pre	20	7.99
SAE-fine		**11.45**

5. Conclusions

In this paper, we investigate the robustness of stacked autoencoder under sparse attack. An objective function which helps improve robustness of stacked autoencoder is proposed. Compared with conventional stacked autoencoder which minimizes training error, our method not only concerns on classification accuracy but also on the robustness. The main idea is to train a stacked autoencoder which makes classification based on majority of features instead of a few important features. We train such stacked autoencoder by minimizing the standard deviation of weight of hidden layers during the fine-tuning stage. With even weight, the importance of features is closed to each other. As a consequence, the adversary has to modify more feature in order to cause misclassification.

We have good performance of improving robustness with training weight evenness in fine-tuning stage. However, weight evenness in pre-training stage cannot improve the robustness effectively. That is because the weight learned in pre-training will be further tuned in fine-tuning stage. Without weight evenness setting during fine-tuning stage, the final learned weight will go back to uneven. There is another option that we train with weight evenness both in pre-training and fine-tuning stages. However, we observe that the gradient is instable when training network with back propagation algorithm. Thus the training is hard to continue. The underlying mechanism is needed to be further explored.

One of the future works is to applied weight evenness to other deep learning methods. When applying weight evenness to other applications, it is worth to study other kind of expressions of this concept. Standard deviation is fast during optimization, but its contribution to robustness is limited. A weight evenness expression which is more efficient and reprehensive should be further investigated.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 615733) and Research Foundation for Talented Scholars of Foshan University (Grant No. gg040996).

References

[1] Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." Machine learning 20.3 (1995): 273-297.

[2] Bengio, Yoshua. "Learning deep architectures for AI." Foundations and trends® in Machine Learning 2.1 (2009): 1-127.

[3] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[4] Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.

[5] He, Kaiming, et al. "Deep residual learning for image Wang, Ke, Janak J. Parekh, and Salvatore J. Stolfo. "Anagram: A content anomaly detector resistant to mimicry attack." International Workshop on Recent Advances in Intrusion Detection. Springer Berlin Heidelberg, 2006. recognition." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.

[6] Girshick, Ross. "Fast r-cnn." Proceedings of the IEEE International Conference on Computer Vision. 2015.

[7] Girshick, Ross. "Fast r-cnn." Proceedings of the IEEE International Conference on Computer Vision. 2015.

[8] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.

[9] Deng, Li, Geoffrey Hinton, and Brian Kingsbury. "New types of deep neural network learning for speech recognition and related applications: An overview." Acoustics, Speech Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." arXiv preprint arXiv:1412.6572 (2014). and Signal Processing (ICASSP), 2013 IEEE International Conference on. IEEE, 2013.

[10] Hinton, Geoffrey, et al. "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups." IEEE Signal Processing Magazine 29.6 (2012): 82-97.

[11] Mikolov, Tomas, et al. "Recurrent neural network based language model." Interspeech. Vol. 2. 2010.

[12] Sundermeyer, Martin, Ralf Schlüter, and Hermann Ney. "LSTM Neural Networks for Language Modeling." Interspeech. 2012.

[13] [13] Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." science 313.5786 (2006): 504-507.

[14] Biggio, Battista, et al. "Security evaluation of biometric authentication systems under real spoofing attacks." IET biometrics 1.1 (2012): 11-24.
[15] Rodrigues, Ricardo N., Lee Luan Ling, and Venu Govindaraju. "Robustness of multimodal biometric fusion methods against spoof attacks." Journal of Visual Languages & Computing 20.3 (2009): 169-179.
[16] Lowd, Daniel, and Christopher Meek. "Good Word Attacks on Statistical Spam Filters." CEAS. 2005.
[17] Nelson, Blaine, et al. "Misleading learners: Co-opting your spam filter." Machine learning in cyber trust. Springer US, 2009. 17-51.
[18] Wang, Ke, Janak J. Parekh, and Salvatore J. Stolfo. "Anagram: A content anomaly detector resistant to mimicry attack." International Workshop on Recent Advances in Intrusion Detection. Springer Berlin Heidelberg, 2006.
[19] Szegedy, Christian, et al. "Intriguing properties of neural networks." arXiv preprint arXiv:1312.6199 (2013).
[20] Papernot, Nicolas, et al. "The limitations of deep learning in adversarial settings." Security and Privacy (EuroS&P), 2016 IEEE European Symposium on. IEEE, 2016.
[21] Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." arXiv preprint arXiv:1412.6572 (2014).
[22] LeCun, Yann, et al. "Handwritten digit recognition with a back-propagation network." Advances in neural information processing systems. 1990.
[23] Carlini, Nicholas, and David Wagner. "Towards evaluating the robustness of neural networks." arXiv preprint arXiv:1608.04644 (2016).
[24] Demontis, Ambra, et al. "On Security and Sparsity of Linear Classifiers for Adversarial Settings." Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR). Springer International Publishing, 2016.
[25] Kołcz, Aleksander, and Choon Hui Teo. "Feature weighting for improved classifier robustness." CEAS'09: sixth conference on email and anti-spam. 2009.
[26] Biggio, Battista, Giorgio Fumera, and Fabio Roli. "Multiple classifier systems for robust classifier design in adversarial environments." International Journal of Machine Learning and Cybernetics 1.1-4 (2010): 27-41.
[27] Zhang, Fei, et al. "Adversarial feature selection against evasion attacks." IEEE transactions on cybernetics 46.3 (2016): 766-777.
[28] Al-Rfou, Rami, et al. "Theano: A Python framework for fast computation of mathematical expressions." arXiv preprint arXiv:1605.02688 (2016).
[29] Chan, Patrick P. K., et al. "Data sanitization against adversarial label contamination based on data complexity." International Journal of Machine Learning & Cybernetics (2017):1-14.
[30] Biggio, Battista, et al. One-and-a-Half-Class Multiple Classifier Systems for Secure Learning Against Evasion Attacks at Test Time. Multiple Classifier Systems. 2015:168-180.