Robust Representations in Deep Learning

Shu Liu
The Computational and Data Science PhD Program)
Middle Tennessee State University
Murfreesboro, TN 37132, USA
email: sl6b@mtmail.mtsu.edu

Qiang Wu
Department of Mathematical Sciences
Middle Tennessee State University
Murfreesboro, TN 37132, USA
email: qwu@mtsu.edu

***Abstract*—Deep neural networks are playing increasing roles in machine learning and artificial intelligence to handle complicated data. The performance of deep neural networks depends highly on the network architecture and the loss function. While the most common choice for loss function is the squared loss for regression analysis it is known to be sensitive to outliers and adversarial samples. To improve the robustness, we introduce the use of the correntropy loss to the implementation of deep neural networks. We further split the neural network architecture into a feature extraction component and function evaluation component and design four two-stage algorithms to study which component is more impacted by the use of the robust loss. The applications in several real data sets indicates that the robust deep neural networks can efficiently generate robust representations of complicated data and the two-stage algorithms are consistently more powerful than their one-stage counterparts.**

***Index Terms*—deep neural network, LSTM, correntropy loss, robustness**

I. Introduction

The history of artificial neural network dates back at least to the perceptron invented by Rosenblatt [1]. After more than half a century's development, artificial neural networks are playing increasing roles in modern machine learning and artificial intelligence applications. Although it has been proved that the feed-forward neural network with a single hidden layer and sigmoid activation function can approximate any arbitrarily complex continuous mapping with arbitrary precision [2]–[4], more and more evidence shows that deep neural networks could more powerful [5]–[7]. In the past decade, along with the fast development of hardwares and computational power, deep neural networks have been successfully applied to computer vision, speech and audio recognition, language processing,customer relationship management, and many other fields.

The performance of deep neural networks highly depends on the network architecture and the loss function. In the context of supervised learning, three main neural network architectures are popularly used. The fully connected neural networks, the convolutional neural networks, and the recurrent neural networks. While the fully connected neural networks could be more widely applied to any structured data set, convolutional neural networks have been shown powerful for image analysis and computer vision, and recurrent neural networks have been successfully used in time series data, such as speech recognition and natural language processing. Regarding the loss functions, the least square loss and cross-entropy loss are commonly used for regression analysis and classification tasks, respectively.

Robustness concerns may arise in practice when the data is contaminated by outliers. For instance, Rare body poses in human pose estimation, unlikely facial point position in facial landmark detection, imprecise ground-truth annotations,and label misspecification all may result abnormal samples in image processing, outliers may present in financial data due to heavy tailed distributions, and system shock could produce extreme and erratic values in signal processing. In these situations, there are needs to develop deep learning robust approaches because least square loss and cross entropy are well known to be unrobust and sensitive to outliers. Some efforts have been done in the literature, e.g., [8]–[10]. While there are multiple ways to promote algorithm robustness, the most common approach is to adopt a robust loss to train the neural networks and typical examples include the Huber's loss, the Tukey's biweight loss, the truncated least square loss, the Cauchy loss, and the correntropy loss.

In this paper, we propose to build robust deep neural networks by the correntropy loss. We will not only verify its effectiveness, but also thoroughly explore where robustness comes from. To be precise, we recall that a deep neural network is usually regarded as the combination of two components, the feature extraction component and the function evaluation component. We are particularly interested in the impact of the robust loss on these two components and will evaluate if both components are impacted or one is more impacted than the other. In order to make fair comparisons, we design two-stage algorithms and conduct a comparative study on several real world applications. The results indicate two surprising findings: (1) While the robust loss may impact both components, it seems the feature extraction part is more impacted. In other words, robust deep neural networks incline to produce more robust feature representations. (2) The two stage implementation of deep neural networks are always more powerful than the one stage approaches, regardless of the loss functions used.

The rest of the paper is organized as follows. In Section II, we introduce the deep neural networks. In Section III, we introduce the two stage algorithms to build deep neural networks. In Section IV, we apply the proposed algorithms to real world applications and present the results. We close with

 ISBN: 978-1-68558-056-8

conclusions and discussions in Section V.

II. Robust deep neural networks

In this section, we introduce two robust deep neural networks, the robust deep feed-forward neural network and the robust long short-term memory neural network.

A. The correntropy loss

Outliers are abnormal or extreme values in the data that significantly deviate from the rest of the observations. The appearance of a small amount of outliers may reduce the ability of statistical inference and hurt the predictive performance of machine learning models. While outlier detection and removal are used sometimes [11] [12], a more common approach for supervised learning is to adopt a robust loss function.

Given a data set $(\boldsymbol{x}_i, y_i), i = 1, \ldots, n$ with $\mathbf{x}_i \in \mathbb{R}^d$ representing the vector of d explanatory variables and y_i the response, the well-known least square method aims to minimize the the mean square error

$$\min_f \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

where $\hat{y}_i = f(\boldsymbol{x}_i)$ is the prediction of the response variable y_i by a hypothetical function f. The minimization process is conducted over a set of hypothesis functions which could be the set of linear functions in the traditional multiple linear regression or the set of nonlinear functions represented by a neural network architecture. A main advantage of the least square method is its optimality when the noise follows a Gaussian distribution while the main criticism is the lack of robustness when the Gaussianity is violated by outliers of heavy tailed distributions.

The use of correntropy loss for robustness has a long history. Its variant forms have been proposed as goodness-of-fit measures in the literature under different terminologies, such as the Welsch's loss [13], the inverted Gaussian loss [14], the exponential squared loss [15], the reflected normal loss [16], and the maximum correntropy criterion [17] [18]. In this paper we adot the form proposed in [19]:

$$\mathcal{L}(y_i, \hat{y}_i) = \sigma^2 \left(1 - \exp\left(\frac{(y_i - \hat{y}_i)^2}{\sigma^2}\right)\right),$$

where $\sigma > 0$ is a tunable parameter that trades off the robustness and fitting errors. A robust regression approach minimizes the mean correntropy loss

$$\min_f \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(y_i, \hat{y}_i).$$

There exist not only numerous empirical evidences in the literature to show the ability of correntropy loss to promote robustness, the theoretical guarantees were also investigated in recent studies [20]–[23].

B. Robust Deep Feed-forward Neural Network

A fully connected feed-forward neural network (FNN) consists of three parts: the input layer, the hidden layers, and the out put layer. The input layer has d neurons, representing the d features of the input data. Mathematically, for an input vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$, the jth node of the input layer is given by $h_j^0 = x_j$. A FNN can have one or multiple hidden layers. It is called a shallow neural network if there is only one hidden layer and a deep neural network if there are two or more hidden layers. As we have already mentioned above, although a shallow neural network has the ability to well approximate arbitrarily complicated functions, there are both empirical and theoretical evidence that deep neural networks are more powerful in real applications. For hidden layers, the value of each neuron is computed from all neurons of the precedent layer by an affine linear mapping and an activation function: let $h_{l,j}$ denote the j-th node of the l-th layer and d_l be the number of neurons in the l-the layer. Then

$$h_{k,j} = a\left(\sum_{j=1}^{d_{l-1}} w_{l,j,k} h_{l-1,j} + b_{l,j}\right),$$

where $w_{l,j,k} \in \mathbb{R}$, $b_{l,j} \in \mathbb{R}$, and a is an activation function. The most popular choices for the activation function include the sigmoid function, the hyperbolic tangent function, and the rectified linear activation function (ReLU). The output layer of an FNN will produce predicted values for the response variable. For a regression analysis with a scalar response variable, the output layer contains one neuron by linear function of the last hidden layer:

$$\hat{y} = \sum_{j=1}^{d_L} w_{L,j} h_{L,j} + b_L,$$

where L denotes the number of hidden layers. Figure 1 shows an example of FNN with two hidden layers and a single output. The number of output neurons can be more than one for vector valued regression analysis or classification problems.

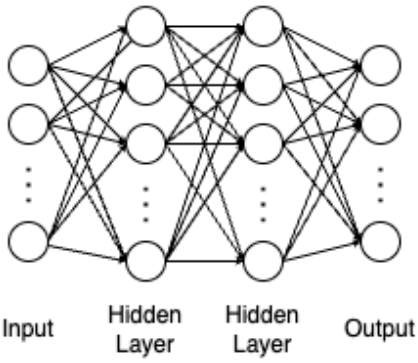

Fig. 1. A deep Feed-forward neural network with two hidden layers

The training of the weight and bias parameters of FNN requires a loss function to measure the error when $\hat{y}_i$ is used to predict the true response value y_i for each observation $\mathbf{x}_i$. In

this paper, we implement the robust deep feed-forward neural network (RFNN) by minimizing the mean correntropy loss.

C. Robust Long Short-Term Memory Neural Network

Long Short-Term Memory Neural Network (LSTM) was first introduced by Hochreiter and Schmidhuber [24]. In 1999, Felix et al. [25] introduced a forget gate mechanism based on Hochreiter and Schmidhuber's work, which enables LSTM to reset its own state to avoid network crashes. Several variant models were proposed since then. LSTM is a special kind of recurrent neural network and has been shown effective for time series analysis, speech recognition, language translation, and natural language processing due to its ability to memorize long and short term information.

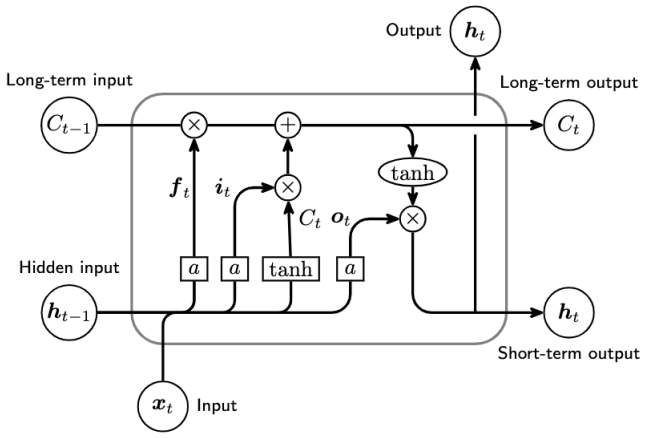

Fig. 2. LSTM network structure

Figure 2 shows the core structure of LSTM. The long term information is stored in the cell states and pass through the entire chain system of LSTM from beginning to end. Three layers will used to decide what information will be removed and what information will be added. The forget gate layer generate $\boldsymbol{f}_t$ of a vector of values between 0 and 1 based on the current input $\boldsymbol{x}_t$ and previous moment output h_{t-1} via affine linear transforms and sigmoid activation function:

$$\boldsymbol{f}_t = a\left(W_f \boldsymbol{x}_t + U_f \boldsymbol{h}_{t-1} + b_f\right)$$

where $W_{\boldsymbol{f}}$ is a matrix of weight coefficients and b_f a sequence of biases. The values of $\boldsymbol{f}_t$ determine the percentage of information in C_{t-1} that are allowed to pass through, in other words, the information $(1-\boldsymbol{f}_t) * C_{t-1}$ will be removed or "forgotten", where $*$ denotes element wise multiplication operator . A tanh layer will produce values representing candidate information :

$$\tilde{C}_t = \tanh\left(W_C \boldsymbol{x}_t + U_C \boldsymbol{h}_{t-1} + b_i\right),$$

and the input gate layer produced $\boldsymbol{i}_t$, again a vector of values between 0 and 1, by

$$\boldsymbol{i}_t = a\left(W_i \boldsymbol{x}_t + U_i \boldsymbol{h}_{t-1} + b_i\right)$$

to decide the percentage of candidate information $\tilde{C}_t$ to be added to the cell state. The cell state is then updated by

$$C_t = \boldsymbol{f}_t * C_{t-1} + \boldsymbol{i}_t * \tilde{C}_t.$$

After the cell state is updated, LSTM will use the output gate will first compute

$$\boldsymbol{o}_t = a(W_o \boldsymbol{x}_t + U_o \boldsymbol{h}_{t-1} + b_o)$$

and then use $\tanh(C_t)$ as weight coefficients to generate the output

$$\boldsymbol{h}_t = \boldsymbol{o}_t * \tanh(C_t),$$

which will be further used to produce the prediction of the response variable. In this paper a linear function

$$\hat{y}_t = \boldsymbol{w}^\top \boldsymbol{h}_t + b$$

is used. Given a sequence of time series $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T$ and corresponding response series $y_1, y_2, \ldots, y_T$, the robust LSTM will minimize the mean correntropy loss

$$\frac{1}{T}\sum_{i=1}^{T} \mathcal{L}(y_t, \hat{y}_t)$$

over the historical period to estimate the network parameters.

III. Two Stage Algorithms

When the data are contaminated by outliers or are skewed and have heavy tails, robust algorithms are supposed to perform better. As we will see in Section IV below, our robust deep learning algorithms are indeed superior as expected when they are applied to real applications with robustness concerns.

The success of deep neural networks have been largely attributed to its ability to extract information from the complicated data. Therefore, it is commonly recognized that a deep neural network can be split into two parts: a feature extraction part and a function evaluation part, where the first part extract relevant features from the input data and second part use the features to build a decision function. As we are able to show the superiority of robust deep learning algorithms, we want to explore further and answer the questions that (1)"whether the robust deep learning algorithms promote robustness of feature extraction and lead to robust representation?" and (2) "which part of the network is more impacted by the use of robust loss?" To answer these questions, we propose a series of two stage algorithms.

For FNN and RFNN, we regard the part from the input to the last hidden layer as the feature extraction process and from the last hidden layer to output as the function evaluation part. We first run FNN and robust FNN to build two neural networks. Then we extract the features and run the linear regression with either least square (LS) approach or the robust regression (RR) with correntropy loss. This leads to four two-stage algorithms: FNN+LS, FNN+RR, RFNN+LS, and RFNN+RR, where the FNN+LS uses the features extracted from FNN and least square regression to predict the response variable, FNN+RR uses the features extracted from FNN and robust regression, RFNN+LS uses the features extracted from RFNN and least

square regression, and RFNN+LS uses the features extracted from RFNN and robust regression. If RFNN+LS outperforms FNN+LS, we are able to conclude that the feature extraction process by RFNN is robust. Otherwise the correntropy loss does not robustify the feature extraction process. On the other hand, if RFNN+RR outperforms RFNN+RR, the correntropy loss plays a role in the function evaluation process.

In LSTM we regards the values in output h_t are the features extracted from the input x_t and previous information. We can similarly design four two-stage algorithms to study the robust representation ability of RLSTM.

IV. Applications

In this section, we apply our algorithms to real-world applications and illustrate their effectiveness.

A. Airfoil Data Set

Airfoil Self-Noise Data [26] is a NASA data set which obtained from a series of aerodynamic and acoustic tests of two and three-dimensional airfoil blade sections conducted in an anechoic wind tunnel. It is a multivariate data set with 5 attributes (Frequency, Angle of attack, Chord length, Free-stream velocity and Suction side displacement thickness) measuring scaled sound pressure level. The data set contains 1503 instances. Figure 3 shows the histogram of the response variable. It is clearly left skewed.

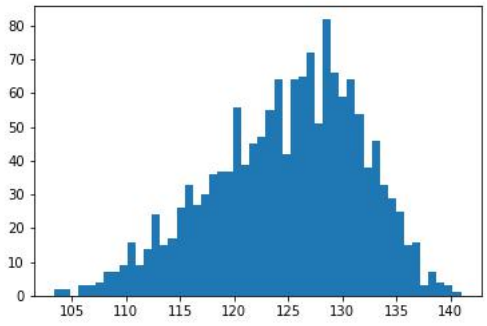

Fig. 3. Histogram of response variable for airfoil data

We randomly sampled 50% (Different split ratios do not significantly affect the final results) of the data as training set and remaining data as test set. We apply FNN, RFNN, and all four two-stage algorithms to build models and predict on the test set. The neural network contains two hidden layers with each hidden layer containing 64 hidden neurons. Tensorflow in Python is used to train the neural network with both the epoch and batch sizes selected as 50. The parameter σ is not sensitive and a value of $\sigma = 10$ is used. The experiments are repeated 50 times. The average mean absolute error (MAE) and the standard error (SE) of all six approaches are reported in Table I.

Firstly, we see that RFNN outperforms FNN, indicating the use of correntropy loss improves the robustness of the neural network estimation. Next, RFNN+LS outperforms FNN+LS and RFNN+RR outperforms FNN+RR, that is, when the same regression approach is used, using features from RFNN is always better. This means that the features extracted by RFNN is more informative and therefore we can claim that the RFNN helps to extract features more robustly. Thirdly, FNN+RR outperforms FNN+LS and RFNN+RR also outperforms RFNN+LS, but the improvement is not significant. This means that once the features have been extracted, further use of robust loss in the regression step does not help much. Lastly, it is surprising to see that FNN+LS outperforms FNN and RFNN+RR outperforms RFNN, indicating that when the same loss function is used, the two-stage algorithms are consistently better than traing the neural network directly. Further more, if we change the loss function in the second regression stage, the two-stage algorithms are still better.

B. Boston Housing Data Set

Boston Housing Data Set contains information collected by the U.S Census Service concerning housing in the area of Boston Massachusetts. It is a multivariate data set with 13 attributes measuring the median value of owner-occupied homes. It contains 506 instances. Figure 4 show the histogram of the home values. We can see outliers on the right end.

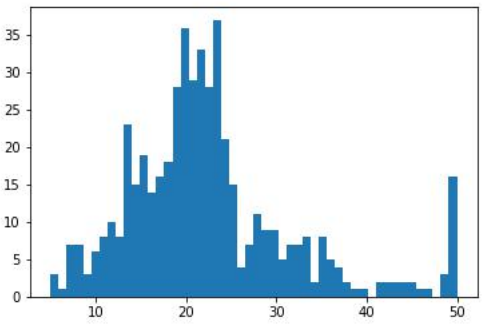

Fig. 4. Histogram of home values in Boston housing data

This data has been build in the sklearn module in Python where the training set and test set have been automatically separated. We merged them together and then randomly sampled 50% as training set and put the remaining data into the test set. The hyperparameters and analysis process are the same as the experiment for Airfoil data. The results are shown in Table I. The findings are very similar to the application in Airfoil data: RFNN is more robust than FNN. The use of correntropy seems play more roles in robust feature extraction while less roles in function evaluation. A follow-up regression stage helps further improve the performance.

C. Agroecosystem Data

This data is collected by Dr. Song Cui at the MTSU Department of Agriculture. It contains carbon, water and energy fluxes of a cool-season dominated pasture ecosystem

 ISBN: 978-1-68558-056-8

for 159 days from September 2, 2016 to May 3, 2017. For each day, there are 48 data points with each data point a summary of the relevant information in half an hour. If the data is complete, there should be 7632 data points in total. But due to power outage or system failure, a small number of data points are missing and the data actually contains 7265 data points. In this analysis, 12 driver variables that measure the radiation, humidity, temperature, wind, and some other features, were used to predict evapotranspiration flux. Outliers due to system shocks can be clearly seen from plot in Figure 5.

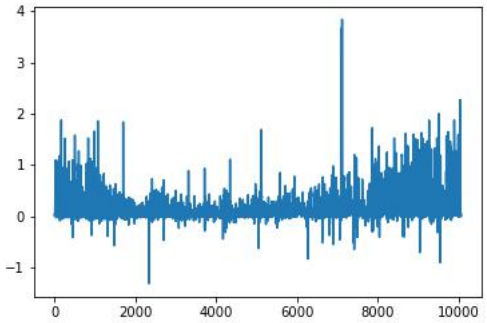

Fig. 5. Plot of evapotranspiration flux

The deep neural network with two hidden layers and 64 hidden neurons for each layer is again used for the analysis. A 50%-50% split is again used for training and test data split. The RFNN used $\sigma = 50$. The experiments are repeated 50 times and the average MAEs are reported for all six approaches in Table I. Similarly, we find the RFNN is able to lead to robust representation of the data and two-stage algorithms are more powerful.

D. CSI 300 Data

Per Wikipedia [27], the CSI 300 is a capitalization-weighted stock market index designed to replicate the performance of the top 300 stocks traded on the Shanghai Stock Exchange and the Shenzhen Stock Exchange. it is a gauge of Chinese stock market. In this experiment, the trading information (opening price, closing price, highest price, lowest price and volume) of CSI 300 from January 3, 2017 to December 29, 2018 (excluding weekends and holidays) used. Figure 6 show the closing prices. Stock prices are typical examples of time series involving sudden changes and abnormal values due to the reaction to government policies, economical indicators, and sentiments.

The analysis aims to predict the closing price based on the opening price, the previous day's opening price, closing price, highest price, lowest price and volume. As the price data can be viewed as time series, LSTM is appropriate. The results by six approaches are reported in Table II. Although a different network architecture is used in this experiment, the findings are still consistent with previous applications. The only difference is that the use of robust regression in

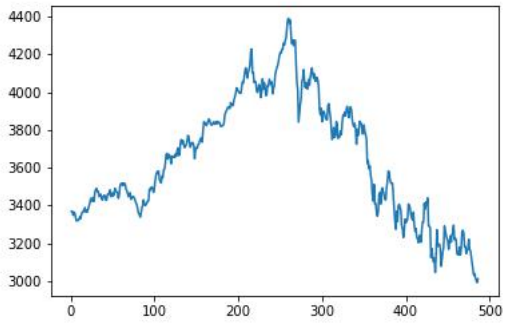

Fig. 6. Histogram of closing prices in CSI 300

TABLE I
MAE ON AIRFOIL AND BOSTON HOUSING DATA

Method	Airfoil	Boston Housing	Agroecosystem
FNN	0.2366 (0.0023)	0.2761 (0.0045)	0.1877 (0.0026)
FNN+LS	0.2235 (0.0016)	0.2719 (0.0040)	0.1720 (0.0007)
FNN+RR	0.2223 (0.0016)	0.2702 (0.0040)	0.1719 (0.0007)
RFNN	0.2279 (0.0018)	0.2706 (0.0035)	0.1779 (0.0014)
RFNN+LS	0.2173 (0.0014)	0.2681 (0.0035)	0.1714 (0.0009)
RFNN+RR	0.2161 (0.0014)	0.2669 (0.0035)	0.1713 (0.0008)

the second stage play more roles, as is evidenced the better performance of LSTM+RR and RLSTM+RR than that of LSTM+LS and RLSTM+LS, respectively.

TABLE II
MAE ON AIU AND CSI300 DATA

Method	CSI300
LSTM	0.2221 (0.0007)
LSTM+LS	0.2133 (0.0007)
LSTM+RR	0.2016 (0.0006)
RLSTM	0.2197 (0.0008)
RLSTM+LS	0.2116 (0.0007)
RLSTM+RR	0.2012 (0.0007)

V. CONCLUSIONS AND FUTURE WORKS

In this paper, we proposed to implement robust deep neural networks by using the correntropy loss and four two-stage algorithms. Simulation studies on four real data applications show that the robust deep neural networks are more efficient to handle data with outliers or skewed. Moreover, the robust deep neural networks are able to efficiently extract more informative features, indicating the entropy loss plays more roles in robust representation of the data.

The superiority of two-stage algorithms is a serendipity. The original motivation of these algorithms is to study how the robust loss plays roles in the network construction process, not for better performance. The simulations surprisingly show that all two-stage algorithms are consistently better than their one-stage counterparts, regardless the loss function used.

In this paper we have focused on the fully connected deep feed-forward neural networks and the LSTM for regression analysis. Convolutional Neural Network (CNN) is another

 ISBN: 978-1-68558-056-8

representative algorithms of deep learning is particularly confidential for its excellent performance in image processing and computer vision. We can similarly develop robust convolutional neural network. However, it seems CNN is more widely used in classification problems while the correntropy loss is more appropriate for regression analysis. So, we have omitted the study of robust CNN in this paper. But the idea of two-stage training is promising and it would be interesting to develop two-stage CNN algorithms with appropriate classification loss functions, such as the cross entropy loss.

ACKNOWLEDGMENT

This work is partially supported by NSF (DMS-2110826).

REFERENCES

[1] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[2] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[3] K.-I. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural networks*, vol. 2, no. 3, pp. 183–192, 1989.

[4] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[5] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press Cambridge, 2016, http://www.deeplearningbook.org.

[8] B. Chen, L. Xing, H. Zhao, N. Zheng, and J. C. Principe, "Generalized correntropy for robust adaptive filtering," *IEEE Transactions on Signal Processing*, vol. 64, no. 13, pp. 3376–3387, 2016.

[9] I. Goodfellow, P. McDaniel, and N. Papernot, "Making machine learning robust against adversarial inputs," *Communications of the ACM*, vol. 61, no. 7, pp. 56–66, 2018.

[10] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rJzIBfZAb

[11] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[12] L. Yin, Q. Wu, and D. Hong, "Statistical methods and software package for medical trend analysis in health rate review process," *J Health Med Inform*, vol. 7, no. 219, 2016.

[13] J. E. Dennis Jr and R. E. Welsch, "Techniques for nonlinear least squares and robust regression," *Communications in Statistics-Simulation and Computation*, vol. 7, no. 4, pp. 345–359, 1978.

[14] K. P. Körding and D. M. Wolpert, "The loss function of sensorimotor learning," *Proceedings of the National Academy of Sciences*, vol. 101, no. 26, pp. 9839–9842, 2004.

[15] X. Wang, Y. Jiang, M. Huang, and H. Zhang, "Robust variable selection with exponential squared loss," *Journal of the American Statistical Association*, vol. 108, no. 502, pp. 632–643, 2013.

[16] F. A. Spiring, "The reflected normal loss function," *Canadian Journal of Statistics*, vol. 21, no. 3, pp. 321–330, 1993.

[17] W. Liu, P. P. Pokharel, and J. C. Principe, "Correntropy: properties and applications in non-Gaussian signal processing," *IEEE Transactions on Signal Processing*, vol. 55, no. 11, pp. 5286–5298, 2007.

[18] J. C. Principe, *Information Theoretic Learning: Renyi's Entropy and Kernel Perspectives*. Springer Science & Business Media, 2010.

[19] Y. Feng, X. Huang, L. Shi, Y. Yang, J. A. Suykens *et al.*, "Learning with the maximum correntropy criterion induced losses for regression." *J. Mach. Learn. Res.*, vol. 16, no. 30, pp. 993–1034, 2015.

[20] Y. Feng, J. Fan, and J. A. Suykens, "A statistical learning approach to modal regression," *Journal of Machine Learning Research*, vol. 21, no. 2, pp. 1–35, 2020.

[21] Y. Feng and Y. Ying, "Learning with correntropy-induced losses for regression with mixture of symmetric stable noise," *Applied and Computational Harmonic Analysis*, vol. 48, no. 2, pp. 795–810, 2020.

[22] Y. Feng and Q. Wu, "Learning under $(1 + \epsilon)$-moment conditions," *Applied and Computational Harmonic Analysis*, vol. 49, no. 2, pp. 495–520, 2020.

[23] ——, "A framework of learning through empirical gain maximization," *Neural Computation*, vol. 33, no. 6, pp. 1656–1697, 2021.

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[25] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[26] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[27] Wikipedia contributors, "CSI 300 index," accessed December 11, 2022. [Online]. Available: https://en.wikipedia.org/wiki/CSI_300_Index

 ISBN: 978-1-68558-056-8