Noise Stable Feed-Forward Neural Networks

A. A. Obeidat
Al-balqa applied university
Al-Hussun university college
Huson, Jordan
e-mail: atefob@hotmail.com

M. H. Essai
Novosibirsk state technical university
Novosibirsk, Russia
e-mail: mhe_2003@yahoo.com

Abstract[1]

The ability of neural networks to closely approximate unknown functions to any degree of desired accuracy has generated considerable demand for neural network research in many fields. The attractiveness of NN research stems from researchers' need to approximate models without having a prior knowledge about the true underlying function, so NNs are known as universal function approximator. All algorithms used to train NNs minimize a mean square error cost function, which is not robust in the presence of large noise such as outliers. Robust statistics introduced various techniques, for estimating the parameters of a parametric model which dealing with deviations from idealized assumptions. We will focus in this paper on one popular robust technique, that is called M-estimators, to minimize the influence of noise on the accuracy of training artificial feed-forward neural networks. We used M-estimators in order to make NN training more efficient and robust. We report simulation results to analyze and evaluate the electiveness of this approach.

1. Introduction

An important, typical problem in mathematic parameter estimation is to find a model that fits a given set of training data. To find the best fitting model, it is necessary to define a merit function that measures the agreement between the data and the model. Mean squared error (MSE) is the preferred measure in many data modeling techniques. Tradition and ease of computation account for the popularity of (MSE).

Supervised neural networks learning algorithms that use mean squared error (MSE) cost function make implicit assumptions such as normality and independence about the error. These assumptions are at best an approximation to reality. One challenge to these assumptions is the occurrence of gross errors which usually appear as outliers. It has been noted, however, that the occurrence of gross errors in routine data ranges from 1% to 10% [8].

Robust statistics have recently emerged as a family of theories and techniques for estimating the parameters of a parametric model while dealing with deviations from idealized assumptions. Examples of deviations include the contamination of data by gross errors, rounding and grouping errors, and departure from an assumed sample distribution. Gross errors or outliers are data severely deviating from the pattern set by the majority of the data. This type of error usually occurs due to mistakes in copying or computation. They can also be due to part of the data not fitting the same model, as in the case of data with multiple clusters. Gross errors are often the most dangerous type of errors. In fact, a single outlier can completely spoil the least squares estimate, causing it to break down. Because of this, trying to detect outliers by thresholding on the residual error will not work. Throwing away one datum at a time and doing least squares on the remaining subset also does not work when more than one outlier is present. To overcome the possibility of presence of outliers, many robust estimators are being used recently such as M-estimators (Maximum-likelihood estimators) [9], W-estimators [11], L-estimators (Linear combination of order statistics) [12].

Nowadays, one popular data modeling technique is artificial neural networks (NN's), which have found many applications in all kinds of fields [1]. It is widely known that feed-forward neural networks are universal approximators [4], [18]. These networks have emerged from the continuous efforts to understand and mimic how the human brain and the nervous system work. Mean squared error (MSE) is also the preferred measure in many artificial neural network architectures and learning algorithms, such as perceptron, back-propagation [2], quickprop [3] and radial basis function (RBF) net [5]. Robust statistics, however, are not widely known in the NN community. The objective of this paper is to apply M-estimators to minimize the influence of gross errors on the accuracy of NN models. The neural network is thus hopefully able to identify inliers (good data) from outlying data such that the correct data model is

Proceedings of the 12th international workshop on computer science and information technologies CSIT'2010, Moscow – Saint-Petersburg, Russia, 2010

estimated. There have been previous efforts in that regard [7], [8], [9], [10], [19]. In this paper we give an extensive study, in terms of accuracy of two M-estimators as robust (noise stable) cost functions with two famous neural network training functions that updates neural networks weights and bias, Levenberg-Marquardt backpropagation (TRAINLM), and Conjugate gradient backpropagation with Fletcher-Reeves updates (TRAINCGF).

2. M-Estimators and its influence functions

M-estimators, is one popular robust technique which correspond to the maximum likelihood type estimate. They generalize straightforwardly to multi-parameter problems [9]. Let r_i be the residual of the it datum, i.e. the difference between the i^{th} observation and its fitted value. The standard least-squares method tries to minimize $\sum_i r_i^2$ which is unstable if there are outliers present in the data. Outlying data give an effect so strong in the minimization that the parameters thus estimated are distorted. The M-estimators try to reduce the effect of outliers by replacing the squared residuals r_i^2 by another function of the residuals, yielding [15], [20]

$$\min \sum_i \rho(r_i), \qquad (1)$$

where $\rho(r_{i.})$ is a symmetric, positive-definite function with a unique minimum at zero, and is chosen to be less increasing than square. Instead of solving directly this problem, we can implement it as an iterated reweighed least-squares one. Now let us see how.

Let $P=[p_1,\ldots,p_m]^T$ be the parameter vector to be estimated. The M-estimator of P based on the function $\rho(.)$ is the vector P which is the solution of the following m equations:

$$\sum_i \psi(r_i)\frac{\partial r_i}{\partial p_j}=0, \quad \text{for } j=1,\ldots,m, \qquad (2)$$

where the derivative $\psi(r_i)=d\rho(r_i)/dr_i$ is called the influence function. If now we define a weight function

$$\omega(r_i)=\frac{\psi(r_i)}{r_i}. \qquad (3)$$

The equation (2) becomes

$$\sum_i \omega(r_i)r_i\frac{\partial r_i}{\partial p_j}=0 \text{ for } j=1,\ldots,m. \qquad (4)$$

This is exactly the system of equations that we obtain if we solve the following iterated reweighed least-squares problem

$$\min \sum_i \omega(r_i^{(k-1)})r_i^2, \qquad (5)$$

where the subscript (k) indicates the iteration number. The weight $\omega(r_i^{(k-1)})$ should be computed after each iteration in order to be used in the next iteration.

Table 1. Used M-estimators

Type	$\rho(r)$	$\psi(r)$	$\omega(r)$
Least Squares (L2)	$r^2/2$	r	1
Least Absolute (L1)	$\lvert r\rvert$	Sgn(r)	$\frac{1}{\lvert r\rvert}$
Least Mean Log Of Squares(LMLS)	$\log(1+\frac{1}{2}r^2)$	$\frac{r}{1+\frac{1}{2}r^2}$	$\frac{1}{1+\frac{1}{2}r^2}$

The influence function measures the influence of a datum on the value of the parameter estimate. For example, for the least-squares with $\rho(x)=x^2/2$ the influence function is $\psi(x)=x$, that is, the influence of a datum on the estimate increases linearly with the size of its error, which confirms the non-robustness of the least-squares estimate. When an estimator is robust, it may be inferred that the influence of any single observation (datum) is insufficient to yield any significant offset [13]. There are some important constraints that robust M-estimators should meet, these constraints are: 1 – Is to have a bounded influence function. 2 – Naturally the requirement of the robust estimator to be unique. This implies that the objective function of parameter vector P to be minimized should have a unique minimum. This requires that the individual ρ-function is convex in variable P. This is necessary because only requiring a ρ-function to have a unique minimum is not sufficient. This is the case with maxima when considering mixture distribution; the sum of unimodal probability distributions is very often multimodal. The convexity constraint is equivalent to imposing that $\partial^2\rho(.)/\partial P^2$ is non-negative definite.

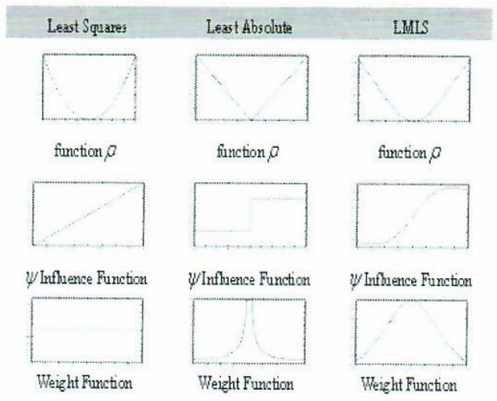

Fig. 1. Graphical representation of used M-estimators.

3 – This is a practical requirement, whenever $\partial^2\rho(.)/\partial P^2$ is singular, the objective should have a gradient, i.e. $\partial\rho(.)/\partial P \neq 0$. This avoids having to search through the complete parameter space. Table (1) lists used M-estimators [15] and their influence functions are graphically illustrated in Fig. 1.

3. Simulation results

In this section, the proper performance of neural networks trained with the robust (M-estimators) is demonstrated in several situations. It was tested to approximate the following function.

$$y = \mathrm{sinc}(x) = \begin{cases} 1, & x = 0 \\ \dfrac{\sin(\pi x)}{\pi x}, & x \neq 0 \end{cases} \qquad (6)$$

We shall compare between robust learning algorithms based on the minimization of M-estimators-based cost functions, and a non-robust learning algorithms based on the minimization of (MSE) cost function.

The neural network architecture considered is a two-layer feed-forward with ten hidden neurons. A total of 500 training patterns were generated by sampling the independent variable in the range [-5, 5], and using (6) to calculate the dependent variable.

We study the training of these neural networks in four cases:

Case A, pure data without any disturbance, in order to get other variants beside the most popular MSE cost function.

Case B, Neural networks trained with high-quality data corrupted with small Gaussian noise: $G_2 \sim N\,(0,0.1)$.

Cace C, Neural networks trained with data corrupted with Gaussian noise, $G2$, in addition to high value random outliers of the form: $H_1 \sim N\,(+15,2)$, $H_2 \sim N\,(-20,3)$, $H3 \sim N\,(+30,1.5)$, $H_4 \sim N\,(-12,3)$.

The data perturbation used in this case is as follows:

$$\mathrm{Data} = (1 - \varepsilon\%)\,\mathrm{G}_2 + \varepsilon\%(\mathrm{H}_1 + H_2 + H_3 + H_4)$$

The outliers were introduced in the data in several percentages: case C.1. with $\varepsilon\%=0.05$; case C.2. with $\varepsilon\%=0.45$.

Case D, Neural networks trained with 51% of the data corrupted with Gaussian noise $G_2 \sim N\,(0,0.1)$; and the remaining 49% of the data substituted by background noise, uniformly distributed.

5. Results

To compare the above four performances, it is necessary to use only one criteria. Therefore, we used the root mean square error (RMSE) of each model.

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} (t_i - y_i)^2}{N}}$$

Where the target t_i is the actual value of the function at x_i and y_i is the output of the network given x_i as its input.

Each of four data sets is trained with both MSE and two M-estimators for 500 epochs, and by using two different training functions (TRAINCGF) and (TRAINLM) within MatLab environment. The results presented below are the average response of trainings. This was done to take into account the different initial values of weights and bias at the beginning of each training. The results are summarized in Table (2), Table (3).

Table 2. RMSE values of networks trained with MSE, L1, LMLS, cost functions and TRAINCGF training function

Training cases	RMSE		
	MSE	L1	LMLS
Case A	0.0064	0.0265	0.0126
Case B	0.0256	0.0348	0.0256
Case C1	1.0490	0.0420	0.0275
Case C2	3.1266	0.0606	0.0566
Case D	0.4056	0.3364	0.3817

Case A: Simulation results using the first data set are presented in Table (2). From these results we observe that the best neural network performance is affected by MSE cost function with RMSE= 0.0064, followed by LMLS with RMSE= 0.0126, and L1 with RMSE=0.0265.

Case B: when Gaussian noise is added to the training data, the RMSE for both MSE and LMLS are similar= 0.0256, followed by L1 with RMSE=0.0348. Although the MSE method is optimal theoretically with respect to Gaussian noise, the LMLS and L1 cost functions performs equally well.

Case C: In the third experiment, the networks are trained using Gaussian-noised data set in addition to high value random outliers at two different percentages. The tabulated results in Table (2) show that the LMLS and L1 cost functions are superior, and have RMSE vales smaller than that corresponding with MSE.

Case D: in this case with background noised training data set, we can note clearly from tabulated RMSE values in

Table (2) that all networks responses based on three cost functions yield poor performance.

Again we will repeat the above training cases but this time using different training function that is called (TRAINLM), the experiments RMSE values in Table(3). From tabulated results we can note clearly the bad performances of the proposed cost functions with this training function except the most noticed response with case A (pure training data set). And some extent good performances with case B (high-quality data corrupted with small Gaussian noise).

Table 3. RMSE values of networks trained with MSE, L1, LMLS, cost functions and TRAINLM training function on training cases

Training cases	RMSE		
	MSE	L1	LMLS
Case A	6.7558e-005	0.0037	3.3664e-005
Case B	0.0284	0.0323	0.0283
Case C1	2.8118	0.4518	0.4977
Case C2	3.8073	1.4144	1.2346
Case D	0.5186	0.4070	0.4148

4. Conclusion

We have introduced noise stable feed-forward neural networks or in other correct words noise stable backpropagation learning algorithm that based on robust cost functions. This robust learning algorithm possesses important properties such as robustness against large noise (outliers), easy adaptation to most neural networks learning algorithms, and high breakdown point. The future direction of this work is aimed toward improving the performance of these estimators especially in the case of data with unstructured background noise, and to test it in a real industrial process. We strongly recommend for those using feed-forward neural networks with data collected from real life systems don't use MSE as cost function.

References

1. Hecht-Nielsen R. "Neurocomputing. Reading", MA: Addison-Wesley, 1990.
2. Rumelhart D. E., McClelland J. L., and the PDP research group. "Parallel Distributed Processing", vol. 1. Cambridge, MA: MIT Press, 1986.
3. Fahlman S. E. "Faster-learning variations on backpropagalion: An empirical study". In Proc. Connectionist Models Summer School, D. Touretzky, G. Hinton, and T. Sejnowski, Eds., pp. 38–51, 1988.
4. Platt J. "A resource-allocating network for function interpolation,"Neural Computa., vol. 3, pp. 213–225, 1991.
5. Moody J., Darken C. "Fast learning in networks of locally-tuned processing units," Neural Computa., vol. 1, pp. 281–284, 1989.
6. Tollenaere T. "Supersab: Fast adaptive backpropagation with good scaling properties," Neural Networks, vol. 3, pp. 561–573, 1990.
7. Rousseeuw P. J., Leroy A. M. "Robust Regression and Outlier Detection". New York: Wiley, 1987.
8. Hampel F. R., Ronchetti E. M., Rousseeuw P. J., and Stahel W. A., "Robust. Statistics, The Approach Based on Influence Functions". New York: Wiley, 1986.
9. Huber P. J. Robust Statistics. John Wiley & Sons, New York, 1981.
10. Wadsworth H. M., Jr., Ed., "Handbook of Statistical Methods for Engineers and Scientists". New York: McGraw-Hill, 1990.
11. Goodall C. "M-estimators of location: An outline of the theory". In D. Hoaglin, F. Mosteller, and J. W. Turkey, editors, Understanding Robust and Exploratory Data Analysis, pp. 339–403. New York, 1983.
12. Koenker R., Basset G. Jr. "Regression quantiles. Econometrica", 36: 33–50, 1978.
13. William J. J. Rey. "Introduction to Robust and Quasi-Robust Statistical Methods". Springer, Berlin, Heidelberg, 1983.
14. Rumelhart D. E., Hinton G. E., Williams R. J. "Learning Representations by propagating Errors. Nature", 323, October, 9, (533-546), 1986.
15. Zhengyou Zhang. "Parameter Estimation Techniques: A Tutorial with Application to Conic Fitting". October, 1995.
16. Liano K. "Robust error measure for supervised neural network learning with outliers," IEEE Trans. Neural Networks, vol. 7, pp. 246–250, Jan. 1996.
17. Alpha V. "TAO-robust backpropagation learning algorithm". Neural Networks, pp. 1–14, 2005.
18. Hornik, K., "Multilayer Feedforward Networks are Universal Approximators", Neural Networks, Vol. 2,pp 359-366, 1989.
19. Chuang C.-C, Su S.-F, &Hsiao C.-C. "The annealing robust bacpropagation (ARBP) learning algorithm". IEEE Transaction on Neural Networks, 11(5), 1067–1077, 2000.
20. Ahmed M., Farag A. "A neural approach to zoom-lens camera calibration from data with outliers. Image and Vision Computing", 20, 619–630, 2002.