The Normalized Risk-Averting Error Criterion for Avoiding Nonglobal Local Minima in Training Neural Networks☆

James Ting-Ho Lo[a], Yichuan Gui[b], Yun Peng[b]

[a]Department of Mathematics and Statistics
[b]Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County
Baltimore, Maryland 21250, USA

Abstract

The convexification method for data fitting is capable of avoiding nonglobal local minima, but suffers from two shortcomings: The risk-averting error (RAE) criterion grows exponentially as its risk-sensitivity index λ increases, and the existing method of determining λ is often not effective. To eliminate these shortcomings, the normalized RAE (NRAE) is herein proposed. As NRAE is a monotone increasing function of RAE, the region without a nonglobal local minimum of NRAE expands as does that of RAE. However, NRAE does not grow unboundedly as does RAE.

The performances of training with NRAE at a fixed λ are reported. Over a large range of the risk-sensitivity index, such training has a high rate of achieving a global or near global minimum starting with different initial weight vectors of the neural network under training. It is observed that at a large λ, the landscape of the NRAE is rather flat, which slows down the training to a halt. This observation motivates the development of the NRAE-MSE method that exploits the large region of an NRAE without a nonglobal local minimum and takes excursions from time to time for training with the standard mean squared error (MSE) to zero into a global or near global minimum. A number of examples of approximating functions that involve fine features or unevenly-sampled segments are used to test the method. Numerical experiments show that the NRAE-MSE training method has a success rate of 100% in all the testing trials for each example, all starting with randomly selected initial weights. The method is also applied to classifying numerals in the well-known MNIST dataset. The new training method outperforms other methods reported in the literature under the same operating conditions.

Keywords: Neural network, training, convexification, risk-averting error, global optimization, local minimum

1. Introduction

The local minimum problem has plagued the development and application of the neural network approach based on the multilayer perceptron (MLP) and its variants, and has attracted much attention since its inception [1, 2, 4, 7, 8, 14–16, 19]. A promising method to alleviate the problem was proposed in [11, 12]. The method employs a new type of risk-averting error (RAE) criterion that was designed to avoid nonglobal local minima. The RAE is a transformation of the standard mean squared error (MSE) criterion for training the MLP. The transformation convexifies the MSE. More specifically, as a parameter, called risk-sensitivity index λ, of the RAE increases, the convexity region of the RAE expands, thereby creating tunnels or wormholes for a local search method such as the gradient descent, conjugate gradient and quasi-Newton algorithm to avoid nonglobal local minima.

The mentioned RAE $J_\lambda(w)$ transforms the MSE as follows [11]:

$$J_\lambda(w) := \sum_{k=1}^{K} \exp\left(\lambda \left\| y_k - \hat{f}\,(x_k, w)\right\|^2\right) \tag{1}$$

Note that $\lim_{\lambda\to 0} \frac{1}{\lambda} \ln\left[\frac{1}{K} J_\lambda\,(w)\right] = Q\,(w)$. The convexification property confirmed the effectiveness of the adaptive training method reported in [12] in avoiding nonglobal local minima. However, $J_\lambda(w)$ is an exponential function of $\lambda\left\| y_k - \hat{f}\,(x_k, w)\right\|^2$ and causes computer register overflow if the risk-sensitivity index λ is large. Furthermore, the adaptive training method, which starts with a very small λ and increases it gradually to expand the convexity region, is not an existing method of selecting λ.

This motivates the use of the normalized RAE (NRAE)

$$C_\lambda\,(w) := \frac{1}{\lambda} \ln\left[\frac{1}{K} J_\lambda\,(w)\right] \tag{2}$$

which is a strictly increasing function of $J_\lambda(w)$. Therefore, the region in $C_\lambda\,(w)$ that has no nonglobal local minima contains the convexity region of $J_\lambda\,(w)$ and thus expands as does said convexity region. The NRAE does not grow exponentially as λ increases.

☆This material is based upon work supported in part by the National Science Foundation under Grant ECCS1028048, but does not necessarily reflect the position or policy of the Government.

Email addresses: jameslo@umbc.edu (James Ting-Ho Lo), yichgui1@umbc.edu (Yichuan Gui), ypeng@umbc.edu (Yun Peng)

To understand the NRAE $C_\lambda(w)$ at various values of λ, trainings of multilayer perceptrons (MLPs) with $C_\lambda(w)$ at fixed values of λ were performed, and the training performances were monitored. The method of training a neural network with $C_\lambda(w)$ at a fixed λ throughout the training is called the NRAE training method. The method was applied for a number of examples of approximating functions with fine features and unevenly-sampled segments specially designed to test the capability of the NRAE training method and thereby gain understanding of the NRAE $C_\lambda(w)$. For each example of approximating a function, the NRAE training method was used with $C_\lambda(w)$ at each fixed value of λ from a number of large values of λ, a large number of training sessions (or trials) starting with different randomly selected initial weight vectors were carried out, and the performance evolutions of MLPs under training during the training sessions were recorded.

The percentage of the training sessions for each of the numerical examples that are successful is 50% for the risk-sensitivity index λ in the range of 10^6-10^8, 100% in the range of 10^8-10^9, and 75% in the range of 10^9-10^{11}, but fails to work for $\lambda > 10^{11}$. Here, a training session is regarded as successful if $C_\lambda(w)$ and $Q(w)$ of the resultant MLP are less than 10^{-9}. The smaller λ is, the more nonglobal local minima there are. This explains the success rate of 50% for λ in the range of 10^6-10^8. The larger λ is, the greater the region with a nonglobal local minimum is. However, when λ increases, the landscape of $C_\lambda(w)$ gradually "flattens". As λ goes beyond 10^9, the training with $C_\lambda(w)$ often slows to a halt, accounting for the success rate of 75% for λ in the range of 10^9-10^{11}, and the success rate reducing to virtually zero for $\lambda > 10^{11}$.

Although the NRAE training method cannot reach a global minimum with a 100% success rate, experiments show that it was able to bring $C_\lambda(w)$ and the corresponding $Q(w)$ significantly down for $10^6 \le \lambda \le 10^{11}$. This observation motivated the following training method: After the NRAE training is performed for a reasonable number of epochs (or iterations), the training criterion is switched from $C_\lambda(w)$ to the MSE $Q(w)$. If a global or nearly global minimum is reached in this training excursion with the MSE, the training is successful and stopped. Otherwise, the NRAE training resumes from the weight vector that the MSE excursion started with and continues for another reasonable number of iterations to be followed by another excursion with MSE. In short, the method comprises a sequence of cycles, each cycle consisting of an NRAE training step followed by an MSE recursion. This method is called the NRAE-MSE training method. The method was found to succeed on all the training sessions for all the examples at each λ in the range 10^6 - 10^{11} in our numerical experiments. In all the mentioned training sessions, cross-validation was performed to ensure that the MLP trained with the NRAE-MSE training method has a good generalization capability.

Encouraged by such performances of the NRAE-MSE training method, we went ahead to test the method on classifying numerals using the well-known MNIST dataset. Compared with the benchmark results obtained by several commonly used methods on the MNIST dataset under the same training conditions, the performance of the NRAE-MSE training method has better test error rates, showing a better generalization capability, under the same experimental conditions with or without the preprocessing for dimension reduction.

2. NRAE Criterion and Its Derivatives

In this section, we show that the computation of the NRAE and its first-order and second-order derivatives involves only values of manageable magnitudes whatever the risk-sensitivity index λ is.

For notational simplicity, let

$$\begin{aligned} \hat{y}_k(w) &:= \hat{f}(x_k, w) \\ \varepsilon_k(w) &:= y_k - \hat{y}_k(w) \ . \end{aligned}$$

For a vector w, let $S(w) = \arg\max_{k\in\{1,\ldots,K\}} \|\varepsilon_k(w)\|^2$ which set may contain more than one elements if a tie exists, and $M(w) = \min_k \{k | k \in S(w)\}$ which is the smallest index among all values in the set $S(w)$. It follows that

$$\|\varepsilon_k(w)\|^2 \le \left\|\varepsilon_{M(w)}(w)\right\|^2 \ .$$

Let

$$\eta_k(w) := e^{\lambda\left(\|\varepsilon_k(w)\|^2 - \left\|\varepsilon_{M(w)}(w)\right\|^2\right)}$$

then

$$\begin{aligned} C_\lambda(w) &= \frac{1}{\lambda}\ln\left[\frac{1}{K}e^{\lambda\left\|\varepsilon_{M(w)}(w)\right\|^2}\sum_{k=1}^{K}\eta_k(w)\right] \\ &= \frac{1}{\lambda}\ln\frac{1}{K} + \left\|\varepsilon_{M(w)}(w)\right\|^2 + \frac{1}{\lambda}\ln\left[\sum_{k=1}^{K}\eta_k(w)\right]. \end{aligned} \tag{3}$$

Note that the number $|S(w)|$ of elements in $S(w)$ may be greater than one, and

$$\begin{aligned} \eta_k(w) &\le 1 \\ \ln\left[\sum_{k=1}^{K}\eta_k(w)\right] &\le \ln K \, . \end{aligned}$$

Hence

$$\begin{aligned} C_\lambda(w) &\le \frac{1}{\lambda}\ln\frac{1}{K} + \left\|\varepsilon_{M(w)}(w)\right\|^2 + \frac{1}{\lambda}\ln K \\ &= \left\|\varepsilon_{M(w)}(w)\right\|^2 \end{aligned}$$

and the terms in Eq. (3) are bounded by functions independent of λ and no register overflow occurs when λ is chosen very large.

Consider the first-order derivative,

$$\begin{aligned} \frac{\partial C_\lambda(w)}{\partial w_j} &= \frac{1}{\lambda J_\lambda(w)}\frac{\partial J_\lambda(w)}{\partial w_j} \\ &= \frac{1}{\lambda J_\lambda(w)}\left[-2\lambda\sum_{k=1}^{K}e^{\lambda\|\varepsilon_k(w)\|^2}\varepsilon_k^T(w)\frac{\partial\hat{y}_k(w)}{\partial w_j}\right] \\ &= \frac{-2\sum_{k=1}^{K}\eta_k(w)\,\varepsilon_k^T(w)\frac{\partial\hat{y}_k(w)}{\partial w_j}}{\sum_{k=1}^{K}\eta_k(w)} \end{aligned} \tag{4}$$

where

$$\sum_{k=1}^{K} \eta_k(w) \le K$$

$$\left| \sum_{k=1}^{K} \eta_k(w)\, \varepsilon_k^T(w)\, \frac{\partial \hat{y}_k(w)}{\partial w_j} \right| \le \sum_{k=1}^{K} \left| \varepsilon_k^T(w)\, \frac{\partial \hat{y}_k(w)}{\partial w_j} \right|$$

which is independent of λ. The computation of $\partial \hat{y}_k(w)/\partial w_j$ has the similar approach like the backpropagation (BP) [17] algorithm. Hence, both the numerator and denominator of Eq. (4) can be handled without register overflow when λ is chosen very large.

Consider the second order derivative:

$$\frac{\partial^2 C_\lambda(w)}{\partial w_i \partial w_j} = \frac{1}{\lambda J_\lambda(w)} \frac{\partial^2 J_\lambda(w)}{\partial w_i \partial w_j} - \frac{1}{\lambda J_\lambda^2(w)} \frac{\partial J_\lambda(w)}{\partial w_i} \frac{\partial J_\lambda(w)}{\partial w_j}. \quad (5)$$

It is shown in [11] that

$$\frac{\partial^2 J_\lambda(w)}{\partial w_i \partial w_j} = 2\lambda \sum_{k=1}^{K} e^{\lambda \|\varepsilon_k(w)\|^2} \left\{ 2\lambda A_{kij}(w) + B_{kij}(w) - C_{kij}(w) \right\}$$

where

$$A_{kij}(w) := \varepsilon_k^T(w) \frac{\partial \hat{y}_k(w)}{\partial w_i} \frac{\partial \hat{y}_k^T(w)}{\partial w_j} \varepsilon_k(w)$$

$$B_{kij}(w) := \frac{\partial \hat{y}_k^T(w)}{\partial w_i} \frac{\partial \hat{y}_k(w)}{\partial w_j}$$

$$C_{kij}(w) := \varepsilon_k^T(w) \frac{\partial^2 \hat{y}_k(w)}{\partial w_i \partial w_j}$$

are all $N \times N$ matrices, where N is the number of weights. It follows that

$$\frac{1}{\lambda J_\lambda(w)} \frac{\partial^2 J_\lambda(w)}{\partial w_i \partial w_j} = \frac{2 \sum_{k=1}^{K} \eta_k(w) \left\{ 2\lambda A_{kij}(w) + B_{kij}(w) - C_{kij}(w) \right\}}{\sum_{k=1}^{K} \eta_k(w)}.$$

Recalling that

$$\frac{\partial J_\lambda(w)}{\partial w_j} = -2\lambda \sum_{k=1}^{K} e^{\lambda \|\varepsilon_k(w)\|^2} \varepsilon_k^T(w) \frac{\partial \hat{y}_k(w)}{\partial w_j}$$

we obtain

$$\frac{1}{\lambda J_\lambda^2(w)} \frac{\partial J_\lambda(w)}{\partial w_i} \frac{\partial J_\lambda(w)}{\partial w_j} = \frac{4\lambda \left(\sum_{k=1}^{K} \eta_k(w)\, \varepsilon_k^T(w)\, \frac{\partial \hat{y}_k(w)}{\partial w_i} \right)}{\sum_{k=1}^{K} \eta_k(w)} \cdot \frac{\left(\sum_{k=1}^{K} \eta_k(w)\, \varepsilon_k^T(w)\, \frac{\partial \hat{y}_k(w)}{\partial w_j} \right)}{\sum_{k=1}^{K} \eta_k(w)}.$$

Notice that $0 < \eta_k(w) \le 1$. Hence, the Hessian matrix $\left[\partial^2 C_\lambda(w)/\partial w_i \partial w_j \right]$ can be evaluated when λ is chosen very large without causing register overflow in computers.

3. The NRAE-MSE training Method

The NRAE-MSE training method is described in the pseudocode, Algorithm 1. In applying the algorithm, we first select a value of λ in the range 10^6 - 10^{11}, select positive integers L and M, which are to be defined later on, and select an initial weight vector $\mathbf{w}_C^*$ for the MLP under training at random. The weight vectors computed in the NRAE training step and MSE excursion are denoted as $\mathbf{w}_C$ and $\mathbf{w}_Q$, respectively. The method then repeats the following two steps:

1. The NRAE training step: Starting with $\mathbf{w}_C^*$, use $C_\lambda(w)$ with the selected λ to train the MLP for L iterations. Each iteration replaces the current weight vector at the beginning of the iteration with the resultant weight vector as the current weight vector. The current weight vector at the end of the L-th iteration is stored as $\mathbf{w}_C^*$.

2. The MSE excursion: Starting with $\mathbf{w}_C^*$, use $Q(w)$ to train the MLP for M iterations. In the process of the M iterations, if $Q(w)$ is less than or equal to a preset positive number ε, or if a cross-validation test shows that an overfitting of the training data occurs, stop the entire NRAE-MSE training. Otherwise, complete performing the M iterations, store the current weight vector as $\mathbf{w}_Q^*$, and return to the step 1.

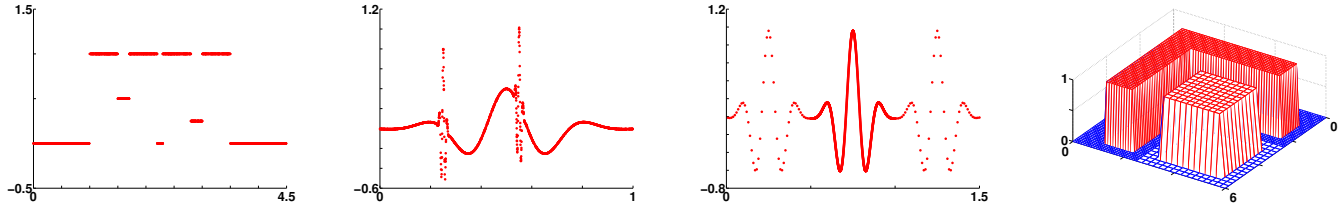
Algorithm 1 NRAE-MSE Training Method

Require: initialize the weight vector $\mathbf{w}$ randomly, choose L, M, $\lambda >> 1$

$\mathbf{w}_C^* \leftarrow \mathbf{w}$
while $Q(\mathbf{w}_Q^*) > \epsilon$ or the overfitting is not detected **do**
  $\mathbf{w}_C(1) \leftarrow \mathbf{w}_C^*$
  for $l = 1$ to L **do** {Step 1: the NRAE training}
    compute the gradient of $C_\lambda(\mathbf{w})$ at $\mathbf{w}_C(l)$
    update $\mathbf{w}_C(l)$ to $\mathbf{w}_C(l+1)$
  end for
  $\mathbf{w}_C^* \leftarrow \mathbf{w}_C(L)$, $\mathbf{w}_Q(1) \leftarrow \mathbf{w}_C^*$
  for $m = 1$ to M **do** {Step 2: the MSE excursion}
    compute the gradient of $Q(\mathbf{w})$ at $\mathbf{w}_Q(m)$
    update $\mathbf{w}_Q(m)$ to $\mathbf{w}_Q(m+1)$
  end for
  $\mathbf{w}_Q^* \leftarrow \mathbf{w}_Q(M)$
end while
return the optimal weight vector $\mathbf{w}_Q^*$

4. Numerical Experiments

In this section, four examples of approximating functions designed to have nonglobal local minima and one example of recognizing handwritten numerals using a real-world dataset, MNIST, are used to demonstrate the effectiveness of the proposed NRAE-based training methods. Several general parameters in training MLPs are chosen with the aid of suggestions in [10]: each synaptic weight in a weight vector is randomly selected from a uniform distribution between $-2.4/F_i$



(a) Three-notch (b) Fine Features (c) Unevenly-sampled Segments (d) Unevenly-sampled Square

Figure 1: Target functions for function approximation examples in Section 4.1. Numbers on the horizontal and vertical axes in each subfigure represent the input and output of the function, respectively. From Fig. 1(a) to Fig. 1(c), red dots denote to the target training data. In Fig. 1(d), different colors (blue or red) are used to distinguish different output values (0 or 1) of the function on vertical axis.

and $2.4/F_i$, where F_i is the number of input neurons of the connected unit; all input and output values defined in the training data are normalized into $[-1, 1]$; the activation function in each training neuron is chosen as the hyperbolic tangent function $\varphi(v) = a\tanh(bv)$, where $a = 1.7159$ and $b = 2/3$.

4.1. Function Approximation

4.1.1. Experimental Design

Four target functions in this experiment are designed to have nonglobal local minima, which are intended to test the capability of the NRAE-based training method for avoiding nonglobal local minima. For approximating a target function, ten different initial weight vectors of an MLP with a certain architecture are randomly chosen. Starting with each such initial weight vector, one standard MSE training, one NRAE training and one NRAE-MSE training session are performed. These 3 training sessions for one and the same initial weight vector are called a training group. The corresponding 2 values of $Q(w)$ of the MLP resulting from the 2 NRAE-based training methods are recorded for comparing to the $Q(w)$ of the MLP resulting from the standard MSE training. In all the training sessions, the derivatives of the MLP are computed by backpropagation, and the MLP weights are updated by the Broyden-Fletcher-Goldfarb-Shanno (BFGS) [3, 5, 6, 18] method.

In order to test the capability of the NRAE-MSE training to tolerate noise in the training data, ten additional training groups are formed for those four function approximation examples with noises added to the target outputs in the training data. The level of the noises added is specified by the signal-to-noise ratio (SNR), which is 10 times the natural logarithm of the ratio of the sum of squares of the target outputs O and the sum of squares of the noises E. Here, noises used in our experiments are the Gaussian white noises, and the SNR is $10\log_{10} 2^2 = 6$dB. The same training strategies used for noiseless data are used for noisy data for each function approximation example.

We use cross-validation for inducing and testing generalization capabilities in all the MSE and NRAE-MSE trainings with noisy data. The size of cross-validation data is one half of that of the training data, and each cross-validation data is randomly selected from the target function without overlapping with the training data.

In each of our sessions of training with the MSE or the NRAE method, the maximum number of training epochs is set equal to 10^6. The NRAE-MSE method may be considered as comprising a sequence of cycles, each cycle consisting of an NRAE training step followed by an MSE training recursion. The maximum number of cycles is set equal to 50 in our experiments. The number of training epochs in the NRAE training step and that in the MSE training excursion in a cycle are denoted by L and M respectively. In our experiments with the NRAE-MSE method, we set $L = 1 \times 10^4$ and $M = 1 \times 10^4$. Therefore, the maximum number of training epochs including those in the NRAE training steps and the MSE recursion is 10^6, which is equal to the number of epochs in the MSE training as well as the NRAE training.

We present our experimental results in this paper for seven large λ values from $10^6 - 10^{11}$, which are 10^6, 10^7, 10^8, 10^9, 10^{10} and 10^{11}. For every example, each λ value is used to perform ten training groups each with one random initial weight vector, one NRAE training session, and one NRAE-MSE training session using the same initial weight vector for the given λ.

Target functions used in function approximation examples are presented in Fig. 1. Definitions of target functions with training and cross-validation data, and MLP architectures for the experiments are described as following:

Three-notch A function with three notches is defined by

$$y = f(x) = \begin{cases} 0 & \text{if } x \in [0, 1.0] \cup [2.2, 2.3] \cup [3.5, 4.5] \\ 0.25 & \text{if } x \in [2.8, 3.0] \\ 0.5 & \text{if } x \in [1.5, 1.7] \\ 1 & \text{otherwise} \end{cases} \tag{6}$$

where $x \in X = [0, 4.5]$. For the training data, input values x_k are selected by randomly sampling 2000 different numbers from a uniform distribution on X, and corresponding output values y_k are computed by Eq. (6). Then, a training dataset of 2000 (x_k, y_k) pairs is obtained. For the cross-validation data, input values x_k are selected by randomly sampling 1000 different numbers from a uniform distribution on X, and corresponding output values y_k are also computed by Eq. (6). Then, a cross-validation dataset of 1000 (x_k, y_k) pairs is obtained. Here, the training dataset and the cross-validation dataset are independent and non-overlapping. MLPs with the 1:16:1 architecture are used in all the training sessions for noiseless or noisy data.

Fine Features A smooth function with two fine features as spikes is defined by

$$y = f(x) = g\left(x, \frac{1}{6}, \frac{1}{2}, \frac{1}{6}\right) + g\left(x, \frac{1}{64}, \frac{1}{4}, \frac{1}{128}\right) + g\left(x, \frac{1}{64}, \frac{11}{20}, \frac{1}{128}\right) \tag{7}$$

where $x \in X = [0, 1]$, and g is defined as

$$g(x, \alpha, \mu, \sigma) = \frac{\alpha}{\sqrt{2\pi}\sigma} \cos\left(\frac{(x-\mu)\pi}{\sigma}\right) \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right). \tag{8}$$

For the training data, input values x_k are selected by randomly sampling 2000 different numbers from a uniform distribution on X, and corresponding output values y_k are computed by Eq. (7). Then, a training dataset of 2000 (x_k, y_k) pairs is obtained. For the cross-validation data, input values x_k are selected by randomly sampling 1000 different numbers from a uniform distribution on X, and corresponding output values y_k are computed by Eq. (7). Then, a cross-validation dataset of 1000 (x_k, y_k) pairs is obtained. Here, the training dataset and the cross-validation dataset are independent and non-overlapping. MLPs with the 1:14:1 architecture are used in all the training sessions for noiseless or noisy data.

Unevenly-sampled Segments A smooth function with two unevenly-sampled segments is defined by

$$y = f(x) = g\left(x, \frac{1}{5}, \frac{1}{4}, \frac{1}{12}\right) + g\left(x, \frac{1}{5}, \frac{3}{4}, \frac{1}{12}\right) + g\left(x, \frac{1}{64}, \frac{5}{4}, \frac{1}{12}\right) \tag{9}$$

where $x \in X = [0, 1.5]$ and g is defined in Eq. (8). For the training data, input values x_k are collected by using 50 grid points from a uniform grid on $[0, 0.5]$, 50 grid points from a uniform grid on $[1.0, 1.5]$, and 2000 grid points from a uniform grid on $(0.5, 1.0)$. Corresponding output values y_k are computed by Eq. (9). These form a training dataset of 2100 (x_k, y_k) input/output pairs. For the cross-validation data, input values x_k are collected by using 25 grid points from a uniform grid on $[0, 0.5]$, 25 grid points from a uniform grid on $[1.0, 1.5]$, and 1000 grid points from a uniform grid on $(0.5, 1.0)$. Corresponding output values y_k are computed by Eq. (9). These form a cross-validation dataset of 1050 (x_k, y_k) input/output pairs. Here, the training dataset and the cross-validation dataset are independent and non-overlapping. MLPs with the 1:12:1 architecture are used in all the training sessions for noiseless or noisy data.

Unevenly-sampled Square A three-dimensional function, which has a letter 'L' shape and an unevenly-sampled square raised from a plane, is defined on $[0, 6] \times [0, 6]$ by

$$z = f(x, y) = \begin{cases} 1 & \text{if } x \in [1.0, 5.5] \text{ and } y \in [1.0, 2.0] \\ 1 & \text{if } x \in [1.0, 2.0] \text{ and } y \in [2.0, 5.5] \\ 1 & \text{if } x \in [3.0, 5.5] \text{ and } y \in [3.0, 5.5] \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

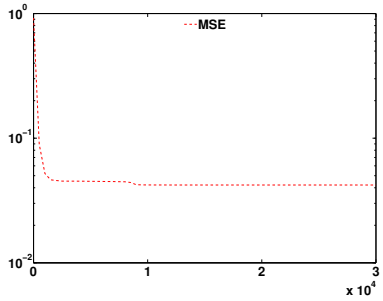
In the training data, input values x_k and y_k are the 289 grid points from the uniform grid on $(2.5, 6] \times (2.5, 6]$ and 2522 grid points from the uniform grid on $[0, 6] \times [0, 6] - (2.5, 6] \times (2.5, 6]$. Corresponding output values z_k are computed by Eq. (10). These form a training dataset of 2811 (x_k, y_k) pairs. In the cross-validation data, input values x_k and y_k are the 144 grid points from the uniform grid on $(2.5, 6] \times (2.5, 6]$ and 1261 grid points from the uniform grid on $[0, 6] \times [0, 6] - (2.5, 6] \times (2.5, 6]$. Corresponding output values z_k are computed by Eq. (10). These form a cross-validation dataset of 1405 (x_k, y_k) pairs. Here, the training dataset and the cross-validation dataset are independent and non-overlapping. MLPs with the 2:6:3:1 architecture are used in all the training sessions for noiseless or noisy data.

4.1.2. Result and Discussion

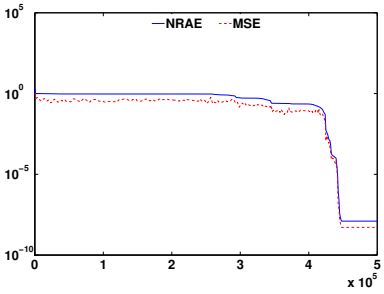
Since we have four function approximation examples, ten sets of initial weights, seven λ values, and datasets with or without noises, we have a total of $4 \times 10 \times 7 \times 2 = 560$ trials for the NRAE-MSE training method, $4 \times 10 \times 7 \times 1 = 280$ trials for the NRAE training method, and $4 \times 10 \times 2 = 80$ trials for the MSE training method. Because of the page limit, we are unable to show all the experimental results here, but we present two comparisons to demonstrate the effectiveness of the NRAE-based training methods as following:

Comparison 1 The three-notch function approximation with noiseless data is chosen as an example to present performances of the NRAE-based training methods over all random initial weight vectors and λ values we tested.

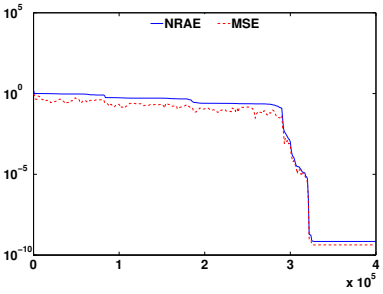
1. Learning curves of the NRAE training sessions presented in Fig. 2 as compared with those of the MSE training sessions show that the NRAE training method with a sufficiently large λ greatly outperforms the MSE training method, and actually reaches a global or near global minimum because the approximation error is nearly zero. Since the three-notch target function is designed to have nonglobal local minima, the learning curves in Fig. 2 indicate that training with the NRAE training criterion with a sufficiently large λ has the capability to avoid nonglobal local minima, while training with MSE does not.
2. In some of our training trials, training with the NRAE criterion fails to reach a global or near global minimum. Table 1 presents all test results of the NRAE and NRAE-MSE training method with different values of λ and initial weights. It shows that in several training sessions with $\lambda = 10^6, 10^7, 10^{10}$ and 10^{11}, training with NRAE is unable to do better than the training with MSE. However, the training with the NRAE-MSE method consistently outperforms the training with MSE in all trials with different sets of initial weights and with all the values of λ under test. It indicates that the NRAE-MSE training method, which switches the training criterion between the NRAE and the MSE, is a significant improvement over the NRAE training method.
3. Ignoring the MSE excursions involved, the NRAE-MSE training is simply an NRAE training. Therefore, the per-



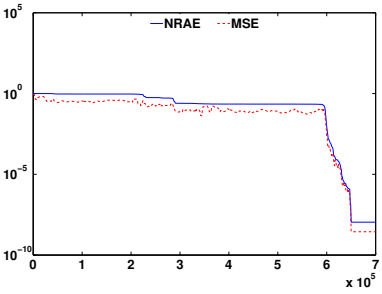
(a) MSE



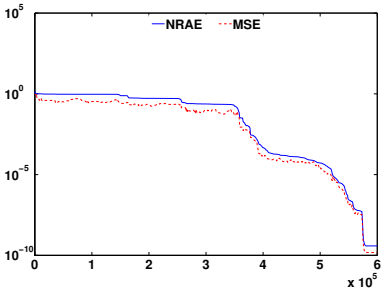
(b) NRAE ($\lambda = 10^6$)



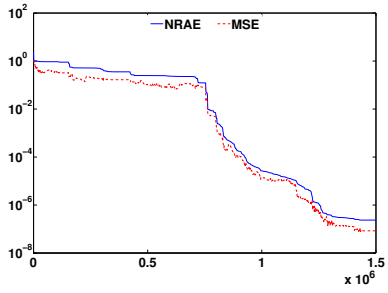
(c) NRAE ($\lambda = 10^7$)



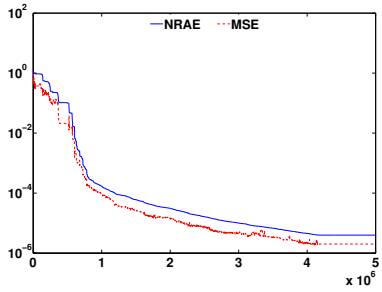
(d) NRAE ($\lambda = 10^8$)



(e) NRAE ($\lambda = 10^9$)



(f) NRAE ($\lambda = 10^{10}$)



(g) NRAE ($\lambda = 10^{11}$)

Figure 2: Learning Curves for the three-notch function approximation with the MSE and NRAE training. For Fig. 2(a), red dash lines represent the MSE training. From Fig. 2(b) to Fig. 2(g), blue solid lines represent the NRAE training and red dash lines are the corresponding curves respect to MSE values. Numbers on the horizontal axis are numbers of training epochs. Numbers on the vertical axis are values of training errors which are converted to the logarithmic numbers with respect to base 10. All training sessions are converged at the end of the shown curves.

Three-notch Function Approximation		Set of Initial Weights									
		1	2	3	4	5	6	7	8	9	10
MSE		1.05×10^{-1}	1.94×10^{-2}	1.11×10^{-4}	1.35×10^{-2}	1.46×10^{-1}	1.01×10^{-1}	7.54×10^{-4}	5.42×10^{-3}	5.41×10^{-3}	8.97×10^{-2}
$\lambda = 10^6$	NRAE	7.14×10^{-8}	8.04×10^{-2}	4.55×10^{-2}	8.36×10^{-7}	9.25×10^{-1}	8.25×10^{-3}	9.36×10^{-8}	7.16×10^{-8}	1.43×10^{-2}	5.23×10^{-9}
	NRAE-MSE	1.47×10^{-5}	3.68×10^{-4}	1.99×10^{-5}	3.25×10^{-4}	2.54×10^{-4}	1.62×10^{-4}	1.24×10^{-4}	9.27×10^{-5}	6.83×10^{-4}	4.85×10^{-4}
$\lambda = 10^7$	NRAE	4.25×10^{-8}	2.86×10^{-2}	3.86×10^{-2}	9.14×10^{-4}	8.04×10^{-7}	4.27×10^{-2}	3.01×10^{-9}	9.02×10^{-8}	9.72×10^{-3}	7.08×10^{-7}
	NRAE-MSE	1.35×10^{-5}	1.57×10^{-5}	1.58×10^{-5}	2.48×10^{-5}	2.06×10^{-4}	1.54×10^{-5}	7.96×10^{-5}	5.05×10^{-5}	2.75×10^{-4}	3.99×10^{-4}
$\lambda = 10^8$	NRAE	4.73×10^{-9}	8.48×10^{-8}	1.95×10^{-8}	1.53×10^{-7}	9.54×10^{-7}	7.64×10^{-7}	5.26×10^{-7}	2.89×10^{-8}	1.74×10^{-7}	6.54×10^{-7}
	NRAE-MSE	4.75×10^{-6}	5.99×10^{-8}	9.00×10^{-9}	7.57×10^{-7}	5.56×10^{-7}	1.69×10^{-6}	7.29×10^{-8}	2.26×10^{-8}	6.74×10^{-8}	9.54×10^{-6}
$\lambda = 10^9$	NRAE	3.52×10^{-8}	7.57×10^{-9}	4.64×10^{-7}	7.23×10^{-7}	5.54×10^{-8}	8.66×10^{-7}	3.80×10^{-9}	4.52×10^{-9}	7.37×10^{-8}	9.36×10^{-7}
	NRAE-MSE	1.86×10^{-7}	6.66×10^{-8}	1.34×10^{-10}	4.83×10^{-8}	4.46×10^{-7}	4.63×10^{-7}	1.63×10^{-8}	9.35×10^{-9}	4.05×10^{-8}	1.35×10^{-7}
$\lambda = 10^{10}$	NRAE	8.43×10^{-8}	1.88×10^{-2}	8.78×10^{-3}	3.65×10^{-6}	1.17×10^{-6}	3.76×10^{-5}	2.01×10^{-3}	7.94×10^{-5}	5.68×10^{-7}	1.94×10^{-6}
	NRAE-MSE	7.33×10^{-8}	1.96×10^{-8}	9.34×10^{-11}	1.24×10^{-8}	7.53×10^{-8}	2.63×10^{-7}	1.74×10^{-8}	4.23×10^{-9}	1.72×10^{-8}	3.78×10^{-8}
$\lambda = 10^{11}$	NRAE	2.00×10^{-6}	6.07×10^{-6}	1.87×10^{-3}	2.16×10^{-4}	3.53×10^{-5}	9.85×10^{-2}	2.02×10^{-5}	4.39×10^{-5}	1.28×10^{-5}	1.27×10^{-7}
	NRAE-MSE	$\mathbf{2.57 \times 10^{-8}}$	$\mathbf{2.16 \times 10^{-8}}$	$\mathbf{5.50 \times 10^{-11}}$	$\mathbf{7.24 \times 10^{-9}}$	$\mathbf{2.54 \times 10^{-8}}$	$\mathbf{1.12 \times 10^{-7}}$	$\mathbf{3.24 \times 10^{-8}}$	$\mathbf{3.21 \times 10^{-9}}$	$\mathbf{9.63 \times 10^{-9}}$	$\mathbf{7.78 \times 10^{-8}}$

Table 1: Training errors of different training methods on the three-notch function approximation with noiseless data in Section 4.1. The row highlighted by the italic font is the MSE training results as a baseline, and the row highlighted by the bold font is the best NRAE-MSE training results among all tests.

formance of the NRAE-MSE training is at least as good as the NRAE training. In Table 1, some NRAE-MSE training errors are larger than the corresponding NRAE training errors when $\lambda = 10^6$, because the total number of training epochs in all the NRAE training steps in the NRAE-MSE training is smaller than the total number of the training epochs completed by the NRAE training method in our experiments. We note that even with a small total number of NRAE training steps, the NRAE-MSE training errors are still lower than their corresponding MSE training errors for all the randomly selected initial weight vectors and all the values of λ tested.

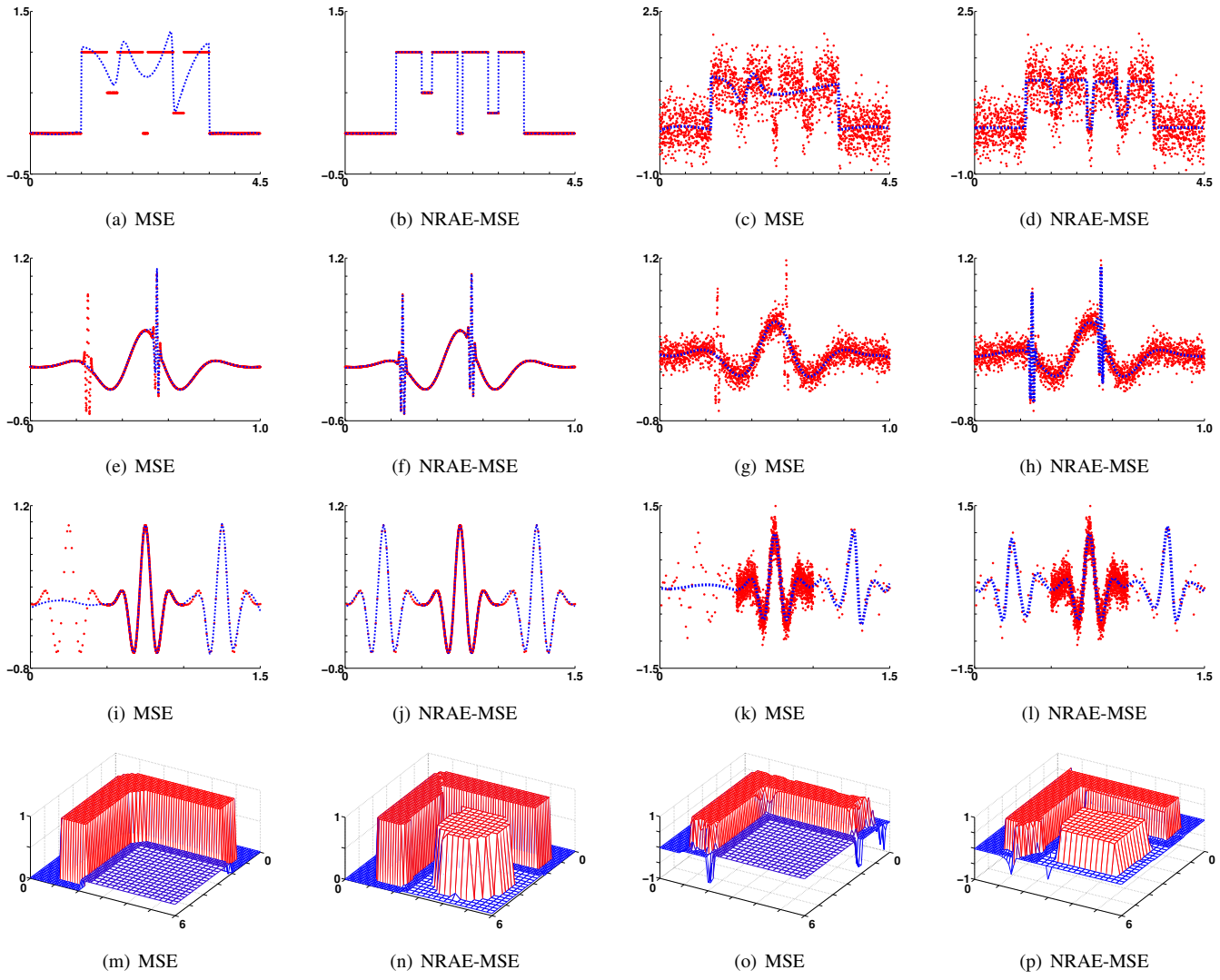
Comparison 2 We choose $\lambda = 10^6$ for all the function approximation examples to compare the NRAE-MSE training method to the standard MSE training method over all random initial weight vectors on training data with or without noise.

1. Experimental results obtained from one of the ten initial weight vectors are illustrated in Fig. 3. It shows that the NRAE-MSE training captures all the significant features and under-sampled segments of the target functions in our experiments with or without noisy data involved, but the MSE training misses all the fine features and under-sampled segments in the target functions in those examples. The results obtained for noisy data are presented in Fig. 3.
2. The NRAE-MSE training method is shown to be able to deal with a high level of noise in the training data. In spite of the noise, the MLP resulting from the NRAE-MSE training still captures the fine features and under-sampled segments of the target function and at the same time maintains a high generalization level.
3. In Fig. 4, it shows that the MLPs obtained by the NRAE-MSE training method outperform all their corresponding MLPs obtained by the MSE training method regardless of the initial weight vectors. Because the training performance of the NRAE-MSE training method is insensitive to the selection of the initial weight vector of the MLP under training, multiple training sessions for the selection of the best resultant MLP is unnecessary.

4.2. Handwritten Digit Recognition using the MNIST Dataset

To test the capability of the NRAE-MSE training method to train a large MLP on a large real-world dataset, it is used to train an MLP for classifying handwritten numerals on the MNIST dataset [9]. The MNIST dataset is commonly used as a benchmark to compare performances of different classifiers, including many neural networks. The MNIST dataset contains 60,000 training samples and 10,000 test samples of handwritten numerals. Each sample has 784 features which are obtained from a 28×28 black and white image. Each feature value is the anti-aliasing normalized gray level of the corresponding pixel in an image.

In our experiments, we test both a transformed and the original MNIST dataset: For the transformed MNIST dataset, we use the principle component analysis (PCA) to reduce the dimensionality of each image in the original MNIST data from 784 to 40 (principle components), and then apply both the standard MSE training method and the NRAE-MSE training method to train a two-layer MLP with the architecture of 40:300:10. Each of the ten output nodes is associated with one of the ten numerals, 1, 2, ..., 9, and 0. For the original MNIST



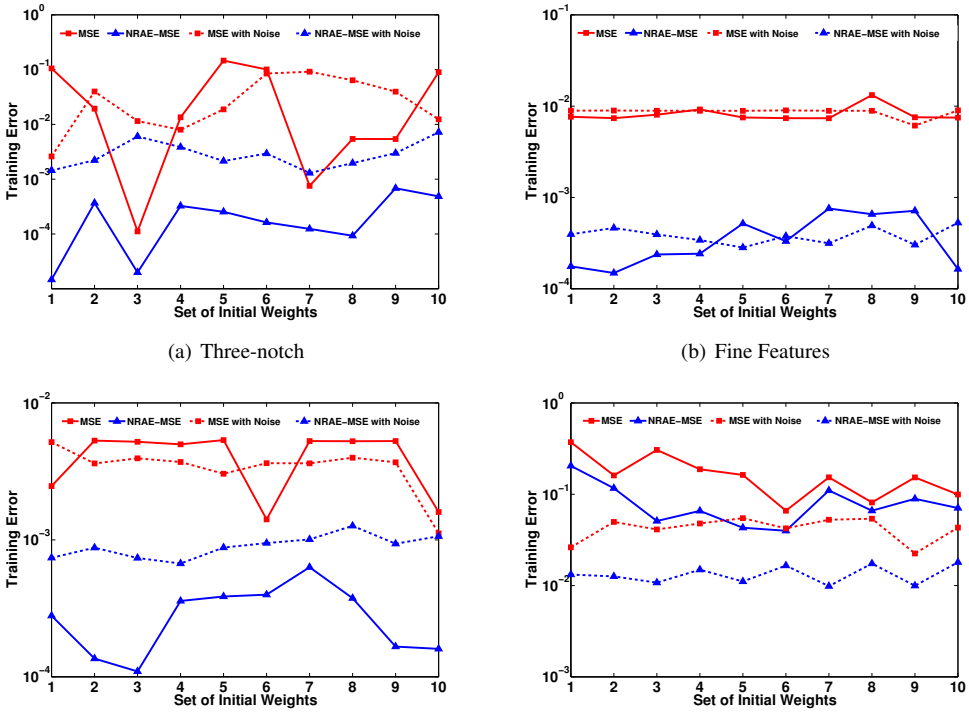
(a) MSE (b) NRAE-MSE (c) MSE (d) NRAE-MSE

(e) MSE (f) NRAE-MSE (g) MSE (h) NRAE-MSE

(i) MSE (j) NRAE-MSE (k) MSE (l) NRAE-MSE

(m) MSE (n) NRAE-MSE (o) MSE (p) NRAE-MSE

Figure 3: Fitting plots for function approximation examples in Section 4.1. For each example, the MSE and NRAE-MSE training method use the same set of initial weights. All NRAE-MSE training sessions presented here are performed with $\lambda = 10^6$. The first two columns describe functions trained with noiseless data, and the last two columns show functions trained with noisy data. Numbers on the horizontal and vertical axes in each subfigure represent the input and output of the function, respectively. From Fig. 3(a) to Fig. 3(l), red dots denote target training samples, and blue dash lines are MLP approximated function plots. From Fig. 3(m) to Fig. 3(p), only MLP approximated function plots are shown by using blue and red colors to distinguish different output values of the functions on vertical axes.



(a) Three-notch

(b) Fine Features

(c) Unevenly-sampled Segments

(d) Unevenly-sampled Square

Figure 4: Training errors of ten different initial weight vectors for function approximation examples in Section 4.1. All NRAE-MSE training sessions presented here are performed with $\lambda = 10^6$. Colors and symbols denote different training methods of the MSE (red square) and NRAE-MSE (blue triangle). Solid and dash lines represent different training sessions with noiseless and noisy data, respectively. In order to clearly show differences between MSE values obtained by the MSE and NRAE-MSE training, actual numbers in all vertical axes are converted to logarithmic numbers with respect to base 10.

dataset, we do the same with a two-layer MLP with the architecture of 784:300:10.

For the training of the MLP, the target value of an output node is 1 if the input is the numeral associated with the node, and is -1 otherwise. After the MLP is trained, the numeral associated with the output node of the MLP outputting the highest value among the ten output nodes is selected as the result of classifying the input image. The classification accuracy of the trained MLP is defined to be the percentage of test images that are correctly classified in the test dataset. We perform the standard gradient descent (GD) optimization method with a momentum term to update weights in a pairwise training mode, which is also called an online training mode, or a sequential training mode. We set the learning rate and the momentum term of the GD method equal to 0.001 and 0.9, respectively. In all the NRAE-MSE training sessions, we set $L = 500$ in the NRAE training and $M = 500$ in the MSE excursion. The maximum number of iterations of the NRAE-MSE training is limited to 10. Therefore, the maximum number of training epochs including both the NRAE training sessions and the MSE excursions is 10^4. We set $\lambda = 10^3, 10^4, 10^5$ and 10^6 to see the effect of the value of λ on the NRAE-MSE training.

Five different initial weight vectors for each chosen value of λ are used in testing. Table 2 presents the best and the worst classification results obtained by the NRAE-MSE training method among all our tests as well as some benchmark results on the MNIST dataset. All NRAE-MSE training sessions apply the early-stopping rule to detect the overfitting and stop the training. The total number of epochs of the NRAE-MSE training in all of our experiments is less than 2000. The experimental results show that the MLP classifier trained by the NRAE-MSE method has a better generalization capability than many benchmark classifiers on the MNIST dataset with or without the PCA applied. Based on our experiments, the NRAE-MSE training method consistently achieves a lower test error rate than the MSE training method in the same settings of MLPs on the MNIST dataset. It confirms that the NRAE-MSE training method has no difficulty in scaling up for a complex real-world problem. More specifically, the method has the ability to avoid nonglobal local minima and maintain a high level of generalization. Moreover, the worst test error rates obtained by the NRAE-MSE training method among all our tests are still better than the MSE training results. We stress that the NRAE-MSE training can provide satisfactory generalization results on any random initial weight vectors, requiring no multiple trials.

5. Conclusion

The normalized risk-averting error (NRAE) criterion is proposed for training neural networks to overcome the local minimum problem, which has plagued the development and application of neural networks. The region in which NRAE does not have a nonglobal local minimum contains the convexity region of the risk-averting error (RAE), which is known to expand as

MNIST Dataset	Classifier	Test Error Rate (%)
With PCA	Quadratic classifier	3.30*
	2-layer MLP 40:300:10, MSE training	3.84
	2-layer MLP 40:300:10, NRAE-MSE training (worst)	**3.02**
	2-layer MLP 40:300:10, NRAE-MSE training (best)	**2.79**
Without PCA	Linear classifier (1-layer MLP)	12.00*
	K-nearest-neighbors, Euclidean (L2)	5.00*
	2-layer MLP 784:300:10, MSE training	4.70*
	2-layer MLP 784:300:10, NRAE-MSE training (worst)	**4.61**
	2-layer MLP 784:300:10, NRAE-MSE training (best)	**4.58**

Table 2: Test error rates of the MNIST dataset for different classifiers. Results marked by asterisks are benchmark results which are available at the MNIST website: http://yann.lecun.com/exdb/mnist/index.html. Results highlighted by bold fonts are achieved by the NRAE-MSE training method. The best and the worst test error rates of the NRAE-MSE training method among all our tests are achieved when $\lambda = 10^6$ and $\lambda = 10^3$, respectively.

its risk-sensitivity index λ increases. The NRAE does not grow exponentially like the way of the risk-averting error (RAE) does as their risk-sensitivity index λ increases. This allows the use of greater values of λ at larger regions in which the NRAE do not have a nonglobal local minimum. Therefore, the NRAE criterion is better suited for overcoming the nonglobal local minimum problem in practice for training neural networks.

Insights to the properties of the NRAE were gained in experiments of training neural networks with the NRAE at each of a large number of values of λ. It was found that as λ increases, the landscape of the NRAE grows flatter, which causes the training slower and sometimes stops short of reaching a global or near global minimum especially when λ exceeds 10^9.

To balance between a greater λ to avoid nonglobal local minima and a smaller λ to avoid excessive slow-down or stop-short of the training, the NRAE-MSE training method was developed. The method trains with an NRAE at a fixed λ, but takes a recursion to train with the MSE from time to time. Whenever the training with the NRAE brings the neural network weights to within an "attraction basin" of a global or near global minimum, the recursion of training with the MSE zeros into it.

The NRAE-MSE training method succeeded each time in all trials of training neural networks with randomly selected initial weights in each of our numerical examples of approximating functions with fine features or unevenly-sampled segments specially designed to create undesirable local minima of the standard MSE criterion that are hard for a training method to escape from. Most real-world applications are not expected to be so "vicious". In our numerical tests, the risk-sensitivity index λ lies in the range 10^6-10^{11}.

To show that the NRAE-MSE training method can be scaled up to a complex real-world problem, it was used to train an MLP for classifying handwritten numerals on the well-known MNIST dataset. The NRAE-MSE training method outperformed the prior methods. It is appropriate to point out that, the NRAE-MSE method succeeded with a single set of randomly selected initial weights. No multiple trials are necessary.

References

[1] E. Aarts, J. Korst, The Neuron, Oxford University Press, 1989.

[2] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, New York, NY, 2006.

[3] C.G. Broyden, The convergence of a class of double-rank minimization algorithms, Journal of the Institute of Mathematics and Its Applications 6 (1970) 76–90.

[4] K.L. Du, M. Swamy, Neural Networks in a Softcomputing Framework, Springer, New York, NY, 2006.

[5] R. Fletcher, A new approach to variable metric algorithms, Computer Journal 13 (1970) 317–322.

[6] D. Goldfarb, A family of variable metric updates derived by variational means, Mathematics of Computation 24 (1970) 23–26.

[7] M.H. Hassoun, Fundamentals of Artificial Neural Networks, MIT Press, Cambridge, Massachusetts, 1995.

[8] S. Haykin, Neural Networks and Learning Machines, Prentice Hall, Upper Saddle River, New Jersey, third edition, 2008.

[9] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, IEEE 86 (1998) 2278–2324.

[10] Y. LeCun, L. Bottou, G.B. Orr, K.R. Muller, Efficient backprop, Lecture Notes in Computer Science in Neural Networks: Tricks of the Trade 1524 (1998) 9–50.

[11] J.T.-H. Lo, Convexification for data fitting, Journal of Global Optimization 46 (2010) 307–315.

[12] J.T.-H. Lo, D. Bassu, An adaptive method of training multilayer perceptrons, in: Proceedings of the International Joint Conference on Neural Networks (IJCNN'01), volume 3, pp. 2013–2018.

[13] J.T-.H. Lo, Y. Gui, Y. Peng, Overcoming the local-minimum problem in training multilayer perceptrons with the NRAE training method, in: Proceedings of the 9th International Conference on Advances in Neural Networks (ISNN'12), volume Part I, pp. 440–447.

[14] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, third edition, Springer, New York, 1999.

[15] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes in C: The Art of Scientific Computing, Cambridge University Press, New York, NY, third edition, 2007.

[16] J.C. Principe, N.R. Euliano, W.C. Lefebvre, Neural and Adaptive Systems: Fundamentals through Simulations, John Wiley and Sons, Inc., New York, 2000.

[17] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (1986) 533–536.

[18] D.F. Shanno, Conditioning of quasi-newton methods for function minimization, Mathematics of Computation 24 (1970) 647–656.

[19] J.M. Zurada, Introduction to Artificial Neural Networks, West Publishing Company, St. Paul, MN, 1992.