Overcoming the Local-Minimum Problem in Training Multilayer Perceptrons by Gradual Deconvexification

James Ting-Ho Lo\textsuperscript{1}, Yichuan Gui\textsuperscript{2}, Yun Peng\textsuperscript{2}
\textsuperscript{1}Department of Mathematics and Statistics
\textsuperscript{2}Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County
Baltimore, Maryland 21250, USA
Email: \{jameslo, yichgui1, ypeng\}@umbc.edu

Abstract—A method of training neural networks using the risk-averting error (RAE) criterion $J_\lambda(w)$, which was presented in IJCNN 2001, has the capability to avoid nonglobal local minima, but suffers from a severe limitation on the magnitude of the risk-sensitivity index $\lambda$. To eliminating the limitation, an improved method using the normalized RAE (NRAE) $C_\lambda(w)$ was proposed in ISNN 2012, but it requires a selection of a proper $\lambda$, whose range may be dependent on the application. A new training method called the gradual deconvexification (GDC) is proposed in this paper. It starts with a very large $\lambda$ and gradually decreases it in the training process until a global minimum of $C_\lambda(w)$ or a good generalization capability is achieved. GDC training method was tested on a large number of numerical examples and produced a very good result in each test.

I. INTRODUCTION

A standard mean squared error (MSE) criterion for training a multilayer perceptron (MLP) to fit a set of input/output pairs, \((x_k, y_k)\), \(k = 1, \ldots, K\), is

$$Q(w) = \frac{1}{K} \sum_{k=1}^{K} \|y_k - \hat{f}(x_k, w)\|^2$$

(1)

where \(\hat{f}(x_k, w)\) denotes the output of an MLP with a weight vector \(w\) in response to input \(x_k\). The input/output pairs, \((x_k, y_k)\), \(k = 1, \ldots, K\), are assumed to satisfy the equation

$$y_k = f(x_k) + \xi_k$$

where \(f\) is an unknown or known function and \(\xi_k\) are random noises or zero. A weight vector for the best fit of the MLP \(f(x, w)\) to the input/output pairs is determined by minimizing \(Q(w)\) by the variation of \(w\). Determining \(w\) for the MLP to fit the input/output pairs is called training the MLP on the training data (i.e., the input/output data).

Training the MLP involves two issues – minimization of \(Q(w)\) and generalization of the MLP on inputs not in the training data. We want to minimize \(Q(w)\) and ensure the generalization capability of the MLP at the same time. These two objectives seem to make training a "multiple-objective problem". Fortunately, the two mentioned issues are separate and independent. In fact, if we don’t know how to find a global minimum, it will be hard to find an MLP that achieve both the objectives. Global minimization of \(Q(w)\) is not a vain issue.

Much effort has been made to avoid nonglobal local minima in training the MLP [1–9]. In most of the research results, the function \(Q(w)\) is not transformed, and a method, algorithm or technique is derived for reducing the fixated \(Q(w)\) by the variation of \(w\). A problem with this approach is that any region not visited by the method, algorithm or technique can be an entirely new world, which may well contain all the global minima and near-global minima at which the MLP has a good generalization capability.

In [10], a method of gradual transforming and convexifying \(Q(w)\) into the following criterion was proposed:

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

(2)

which is called a risk-averting error (RAE) criterion, and \(\lambda\) is the risk-sensitivity index. It is proven in [10] that the convexity region of \(J_\lambda(w)\) expands monotonically as \(\lambda\) increases, thereby reducing nonglobal local minima, and that

$$\lim_{\lambda \to \infty} \frac{1}{\lambda} \ln \left( \frac{1}{K} J_\lambda(w) \right) = Q(w).$$

Convexification using \(J_\lambda(w)\) with these properties is shown to be effective in avoiding nonglobal local minima in [11]. Nevertheless, \(J_\lambda(w)\) is an exponential function of \(\lambda \|y_k - \hat{f}(x_k, w)\|^2\) and is plagued with computer register overflow when \(\lambda\) is large. This motivated the use of the normalized RAE (NRAE)

$$C_\lambda(w) := \frac{1}{\lambda} \ln \left( \frac{1}{K} J_\lambda(w) \right)$$

(3)

leading to the NRAE training method presented in ISNN 2012 [12]. However, its success rate over all the numerical experiments is 50\% for the risk-sensitivity index \(\lambda\) in the range \(10^6 \text{ - } 10^7\), 100\% in the range \(10^8 \text{ - } 10^9\), 75\% in the range \(10^{10} \text{ - } 10^{11}\), and fails to work for \(\lambda > 10^{11}\). The success rate of a method is defined as the ratio of the number of sessions in which the method is successful to the total number of sessions which the method is used. If a training session using a method produces a MLP with a sufficiently small MSE, the method is said to be successful in the session. If the method is successful in all training sessions over all...
numerical examples, it is said to have a 100% success rate on all the examples.

In this paper, a new training method, called the gradual deconvexification (GDC), is proposed to alleviate the difficulty in finding a good value of $\lambda$ for the NRAE training. GDC training method starts with a very large $\lambda$ and gradually decreases it until a global minimum of $C_\lambda (w)$ or a good generalization capability of the MLP is obtained with cross-validation data. Numerical experiments to be presented indicate that GDC training method reaches a global minimum with a 100% success rate on all the examples used in the experiments.

II. NRAE CRITERION AND ITS DERIVATIVES

In order to make sure that any large value can be selected as the initial $\lambda$ in the GDC training method without any numerical error, we demonstrate an analysis of the NRAE criterion, and its first and second order derivatives in this section.

For notational simplicity, let

$$\hat{y}_k (w) := \hat{f} (x_k, w)$$

$$\varepsilon_k (w) := y_k - \hat{y}_k (w).$$

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 $M (w) = \min_k \{ k | k \in S (w) \}$ is the smallest index among all values in the set $S (w)$. It follows that

$$\| \varepsilon_k (w) \|^2 \leq \| \varepsilon_M (w) \|^2.$$ 

Let

$$\eta_k (w) := e^\lambda (\| \varepsilon_k (w) \|^2 - \| \varepsilon_M (w) \|^2)$$

then

$$C_\lambda (w) = \frac{1}{\lambda} \ln \left[ \frac{1}{K} e^{\lambda \| \varepsilon_M (w) \|^2} \sum_{k=1}^{K} \eta_k (w) \right]$$

$$= \frac{1}{\lambda} \ln \frac{1}{K} + \| \varepsilon_M (w) \|^2 + \frac{1}{\lambda} \ln \left[ \sum_{k=1}^{K} \eta_k (w) \right].$$

(4)

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

$$\eta_k (w) \leq 1$$

$$\ln \left[ \sum_{k=1}^{K} \eta_k (w) \right] \leq \ln K.$$

Hence

$$C_\lambda (w) \leq \frac{1}{\lambda} \ln \frac{1}{K} + \| \varepsilon_M (w) \|^2 + \frac{1}{\lambda} \ln K$$

$$= \| \varepsilon_M (w) \|^2$$

and the terms in (4) are bounded by functions independent of $\lambda$ and no register overflow occurs when $\lambda$ is chosen very large.

Consider the first-order derivative,

$$\frac{\partial C_\lambda (w)}{\partial w_j} = \frac{1}{\lambda J_\lambda (w)} \frac{\partial J_\lambda (w)}{\partial w_j}$$

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

$$= -2 \sum_{k=1}^{K} \eta_k (w) \frac{\partial \hat{y}_k (w)}{\partial w_j} \frac{\varepsilon_k (w)}{\sum_{k=1}^{K} \eta_k (w)}$$

(5)

where

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

$$\sum_{k=1}^{K} \eta_k (w) \frac{\partial \hat{y}_k (w)}{\partial w_j} \frac{\varepsilon_k (w)}{\sum_{k=1}^{K} \eta_k (w)} \leq \sum_{k=1}^{K} \frac{\partial \hat{y}_k (w)}{\partial w_j} \frac{\varepsilon_k (w)}{\sum_{k=1}^{K} \eta_k (w)}$$

which is independent of $\lambda$. The computation of $\frac{\partial \hat{y}_k (w)}{\partial w_j}$ has the similar approach like the backpropagation (BP) algorithm. Hence, both the numerator and denominator of (5) can be handled without register overflow when $\lambda$ is chosen very large.

Consider the second order derivative:

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

(6)

It is shown in [10] that

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

where

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

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

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

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

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

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
\[
\frac{1}{\lambda J_{\lambda}^2(w)} \frac{\partial J_{\lambda}(w)}{\partial w_i} \frac{\partial J_{\lambda}(w)}{\partial w_j} = 4\lambda \left( \sum_{k=1}^{K} \eta_k(w) \varepsilon_k^T(w) \frac{\partial \eta_k(w)}{\partial w_i} \right) \sum_{k=1}^{K} \eta_k(w) \left( \sum_{k=1}^{K} \eta_k(w) \varepsilon_k^T(w) \frac{\partial \eta_k(w)}{\partial w_j} \right)
\]
Notice that $0 < \eta_k(w) \leq 1$. Hence, the Hessian matrix $[\partial^2C_{\lambda}(w)/\partial w_i \partial w_j]$ can be evaluated when $\lambda$ is chosen very large without causing register overflow in computers.

III. The Gradual Deconvexification Training Method

GDC training method starts with a very large $\lambda$ (e.g., $10^{20}$) and keeps recording the values of the objective function $C_{\lambda}(w)$, which are denoted as $F_a$ and $F_b$, before and after a preselected number $E$ of training epochs. If $D = |F_a - F_b|$ is less than a threshold $T$, GDC flags the current training condition as stagnant, and performs a deconvexification by reducing the current $\lambda$ to $R\lambda$ with a percentage $R$. After that, the training continues and repeats the deconvexification if necessary until a satisfied training error $\epsilon$ is achieved. If the deconvexification comes down to $\lambda \leq 1$, the training will set $\lambda = 0$, which is actually a MSE training. This is intended to avoid excessive deconvexification steps. GDC algorithm is described in Algorithm 1.

With the GDC, there is no need to select the initial $\lambda$ before the NRAE training, because the deconvexification gradually decreases $\lambda$ to avoid nonglobal minima in the process of training and eventually obtain a global or near-global minimum of the MSE criterion. The decreasing rate of $\lambda$ is controlled by the three parameters $E$, $T$ and $R$. Our numerical experiments show that the GDC training method is not sensitive to these parameters in the sense that they affect only the convergence rate but not the value of $C_{\lambda}(w)$ that is obtained at the end of training. More specifically, as long as the parameters provide a sufficiently small decreasing rate of $\lambda$, GDC training method works well.

IV. Numerical Experiments

The MSE criteria for training MLPs to approximate uniformly sampled functions with fine features and nonuniformly sampled functions with grossly unevenly-sampled segments are known to have nonglobal local minima. Four such training examples are provided in this section to demonstrate the effectiveness of the proposed GDC training method comparing to the standard MSE training. For each task, ten different sets of initial weights of the MLP are randomly chosen to start ten training groups. In each training group, a GDC training session and a MSE training session are performed on the same set of initial weights. For all training sessions, the derivatives of the MLP outputs are computed in the same approach as the BP algorithm, and the trainable weights are updated by the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method. We examine the fitting plots and training errors for all function approximation tasks and compare performances of the GDC and MSE training methods among all training groups. For all GDC training sessions, we compute their corresponding MSE values as training errors and compare them to the results of the MSE training.

In all GDC training sessions, we set $\lambda = 10^{20}$ as the initial value, and $E = 1000$ as the maximum training epochs to check the difference $D$ between two objective function values $F_a$ and $F_b$. We also use $T = 10^{-10}$ as the threshold to decide when a deconvexification is needed to perform. Since all function approximation tasks are not large-scale problems, we set $R = 0.9$ to achieve lower training errors with slower convergence rate. Parameters used in MLPs are chosen based on suggestions in [9]: each synaptic weight in a weight vector is randomly selected from a uniform distribution between $-2.4/N$ and $2.4/N$, where $N$ 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 $\varphi(v) = \tanh(bv)$, where $a = 1.7159$ and $b = 2/3$.

A. Function Approximation

1) Three-notch: A function with three notches is defined by

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

(7)
Fig. 1. Fitting plots for function approximation tasks in Section IV-A. Numbers on horizontal and vertical axes in each subfigure represent the input and output of the function, respectively. From Fig. 1(a) to Fig. 1(f), red dots denote target training samples, and blue dash lines are MLP approximated function plots. On Fig. 1(g) and Fig. 1(h), only MLP approximated function plots are shown by using blue and red colors to distinguish different function values on vertical axes.
where $x \in X = [0, 4.5]$. The input values $x_k$ are obtained by random sampling 2000 non-repeatable numbers from $X$ with a uniform distribution, and the corresponding output values $y_k$ are computed by (7). The training data with 2000 $(x_k, y_k)$ pairs is chosen to perform the three-notch function approximation. MLPs with 1:15:1 architecture are initiated to all training sessions. Approximated function plots of the MSE and NRAE training are presented in Fig. 1(c) and Fig. 1(d). Training errors obtained by ten sets of different initial weights are shown in Fig. 2(b) for the MSE and NRAE trained MLPs.

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

$$y = f(x) = g\left(x, \frac{1}{6}, \frac{1}{2}, \frac{1}{6}\right) + g\left(x, \frac{1}{64}, \frac{1}{4}, \frac{128}{1}\right) + g\left(x, \frac{1}{64}, \frac{11}{20}, \frac{1}{128}\right)$$  \hspace{1cm} (8)

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

$$g(x, \alpha, \mu, \sigma) = \frac{\alpha}{\sqrt{2\pi}\sigma} \cos\left(\frac{(x-\mu)\pi}{\sigma}\right) \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$  \hspace{1cm} (9)

The input values $x_k$ are selected by sampling 2000 numbers from a uniform distributed grid on $X$, and the corresponding output values $y_k$ are computed by (8). The training data with 2000 $(x_k, y_k)$ pairs is chosen to perform the fine features function approximation. MLPs with 1:15:1 architecture are initiated to all training sessions. Approximated function plots of the MSE and NRAE training are presented in Fig. 1(e) and Fig. 1(f). Training errors obtained by ten sets of different initial weights are shown in Fig. 2(c) for the MSE and NRAE trained MLPs.

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

$$y = f(x) = g\left(x, \frac{1}{5}, \frac{1}{4}, \frac{1}{12}\right) + g\left(x, \frac{1}{5}, \frac{3}{4}, \frac{1}{12}\right) + g\left(x, \frac{1}{64}, \frac{5}{4}, \frac{1}{12}\right).$$  \hspace{1cm} (10)

where $x \in X = [0, 1.5]$ and $g$ is defined in (9). The input values $x_k$ are collected by sampling 50 numbers from a uniform distributed grid on $[0, 0.5]$, 50 numbers from a uniform distributed grid on $[1.0, 1.5]$, and 2000 numbers from a uniform distributed grid on $[0.5, 1.0]$. The corresponding output values $y_k$ are computed by (10). The training data with 2100 $(x_k, y_k)$ pairs is chosen to perform the unevenly-sampled segments function approximation. MLPs with 1:12:1 architecture are initiated to all training sessions. Approximated function plots of the MSE and NRAE training are presented in Fig. 1(e) and Fig. 1(f). Training errors obtained by ten sets of different initial weights are shown in Fig. 2(c) for the MSE and NRAE trained MLPs.

4) Unevenly-sampled Square: A three-dimensional function, which has a letter ‘L’ shape and an unevenly-sampled
square raised from a plane, is defined by
\[
z = f(x, y) = \begin{cases} 
1 & \text{if } x \in [1.0, 5.5] \text{ and } y \in [1.0, 2.0] \\
1 & \text{if } x \in [1.0, 2.0] \text{ and } y \in [2.0, 5.5] \\
1 & \text{if } x \in [3.0, 5.5] \text{ and } y \in [3.0, 5.5] \\
0 & \text{otherwise}
\end{cases}
\]
where \(x \in X = [0, 6]\) and \(y \in Y = [0, 6]\). The input values \(x_k\) and \(y_k\) are collected by sampling 289 numbers from an uniform distributed grid on \(X \times Y = (2.5, 6) \times (2.5, 6)\) and 2522 numbers from an uniform distributed grid on \((X - X') \times (Y - Y')\). The corresponding output values \(z_k\) are computed by (11). The training data with 2811 \(((x_k, y_k), z_k)\) pairs is chosen to perform the unevenly-sampled square function approximation. MLPs with 2:9:3:1 architecture are initiated to all training sessions. Approximated function plots of the MSE and NRAE training are presented in Fig. 1(g) and Fig. 1(h). Training errors obtained by ten sets of different initial weights are shown in Fig. 2(d) for the MSE and NRAE trained MLPs.

B. Discussion

Three major advantages of the GDC training method are observed based on our numerical experiments:

1) GDC starts with a very large \(\lambda\), thus it eliminates the need to find an proper initial \(\lambda\) to perform the NRAE training. In our experiments, all GDC training sessions for different function approximation tasks use the same large initial \(\lambda\) and the same settings of \(E, T\) and \(R\). It indicates that those parameters in the GDC training is not sensitive to our training tasks.

2) Since all tested function approximation tasks are intended to have nonglobal local minima, experimental results in Fig. 1 demonstrate that the GDC training have the capability to captures more significant features located on each target function by avoiding nonglobal local minima than the MSE training.

3) Training results in Fig. 2 indicate that the GDC training sessions with ten different sets of initial weights consistently lead all trained MLPs to achieve satisfactory training errors, which are lower than the MSE training achieved. Although the standard deviation of the GDC training is larger than the MSE training over all tests, the performance of the MSE training sessions never outperform the GDC training sessions in our experiments.

V. Conclusion

A method of training MLPs, called the gradual deconvexification (GDC) is proposed that can reach a global minimum of the normalized risk-avering error criterion \(C_\lambda(w)\). If the MLP has enough nodes, \(\min_w C_\lambda(w)\) is virtually zero at a global minimum, implying that \(\min_w Q(w)\) is also virtually zero. If cross-validation data is available and generalization capability of the MLP under training is desired, GDC can be stopped before reaching \(\min_w C_\lambda(w)\). GDC training method has a success rate of 100% in all the experiments that we have performed.