Bernoulli Error Measure Approach to Train Feedforward Artificial Neural Networks for Classification Problems

Mo-yuen Chow  Paul Goode  Alberico Menozzi  Jason Teeter  James P. Thower

Department of Electrical and Computer Engineering
North Carolina State University
Raleigh, NC 27695-7911
USA

Abstract

The training of artificial neural networks usually requires that users define an error measure in order to adapt the network weights to achieve certain performance criteria. This error measure is very important and sometimes essential for achieving satisfactory solutions. Different error measures have been used to train feedforward artificial neural networks, with the mean-square error measure (and its modifications) being the most popular one. In this paper, we will show that the Bernoulli error measure is very suitable for training feedforward artificial neural networks to learn classification problems. We will compare the Bernoulli error measure with the popular mean-square error measure in terms of error surfaces, adaptation rates, and stability regions. The AND and XOR classification problems will be used to illustrate the differences between the Bernoulli error measure and the mean-square error measure.

1. Introduction

The training of artificial neural networks usually requires that users define an error measure in order to adapt the network weights to achieve certain performance criteria. Different error measures have been used to train feedforward artificial neural networks, with the mean-square error measure (and its modifications) being the most popular [1]. The mean-square error measure has been widely used in different neural network applications such as control [2,3], pattern recognition and classification [4,5], among others. In addition to the mean-square error measure family, other measures have been proposed [6] proposed to use a measure that includes a-priori information about the data structure. Different error measures will influence the network learning capability and the network solution. Different problems require different error measures, which include the mean-square, one-norm, infinity-norm, and probability, among others [7]. Indeed, the error measures used in neural network training are very important and sometimes essential for achieving satisfactory solutions.

Artificial neural networks have been used extensively in classification problems. Bernoulli error measure has been used in several areas, mostly in statistics, to distinguish among different classes. [8] has suggested to use the Bernoulli error measure for classification problems. In this paper, we will show that the Bernoulli error measure is very suitable for training feedforward artificial neural networks to learn classification problems. We will compare the Bernoulli error measure with the popular mean-square error measure in terms of error surfaces, adaptation rates, and stability regions. The AND and XOR classification problems will be used to illustrate the differences between the Bernoulli error measure and mean-square error measure. For completeness, Section 2 of this paper will briefly overview the input-output neural network training based on the backpropagation training algorithm. Sections 3 and 4 will discuss the δ-rules for the mean-square error measure and Bernoulli error measure, respectively. Section 5 will compare the learning rates and stability regions of training using the two different error measures of interest. Section 6 will provide the conclusions of this paper.

2. Brief Overview of Learning an Input-Output Mapping with a Feedforward Net

Let \( \mathcal{N} \) be an actual system mapping that maps the input space \( \mathcal{X} \) to the output space \( \mathcal{Y} \):

\[
\mathcal{N} : \mathcal{X} \rightarrow \mathcal{Y},
\]

and let a neural network input-output mapping be \( y(x) = \mathcal{N}(w, x) \in \mathcal{Y}, \ x \in \mathcal{X}, \text{ and } w \in \mathcal{W}, \) the weight space. We want to train the neural network \( \mathcal{N} \) to learn the actual mapping \( \mathcal{N} \) by adjusting \( w \) to minimize a chosen error function \( E(w, x) : \)

\[
\min_{w \in \mathcal{W}} E(w, x) \forall x \in \mathcal{X}.
\]

Normally, the neural network's training termination criteria is either:

\[
|| \mathcal{N}(w, x) - \mathcal{N}(x) || \leq \epsilon_t,
\]

or

\[
|| \Delta w || \leq \epsilon_w.
\]
where $e_t$ is a preset tolerance of the difference between the mappings $\mathfrak{M}$ and $\mathfrak{N}$, and $e_w$ is a preset tolerance of the change in weights during the training session. The values of $e_t$ and $e_w$ are usually small positive numbers.

Usually, the exact mathematical model of the mapping $\mathfrak{M}$ is not known, but the input-output values of $t_p = \mathfrak{M}(x_p)$ can be obtained and are used as training and testing patterns. Let $y_p = \mathfrak{N}(w, x_p)$, and assume that there are $n$ elements in each input pattern, i.e.,

\[ x_p = \{x_{p1}, ..., x_{pn}\}^T, \]

$m$ elements in each output pattern, i.e.,

\[ y_p = \mathfrak{N}(w, x_p) = \{y_{p1}, ..., y_{pm}\}^T, \]

and

\[ t_p = \mathfrak{M}(x_p) = \{t_{p1}, ..., t_{pm}\}^T, \]

with $p = 1, ..., P$, where $P$ is the total number of training patterns used.

Define $e_p = |y_p - t_p|$ as an error measure of the difference between $\mathfrak{N}(w, x_p)$ and $\mathfrak{M}(x_p)$ with inputs $x_p$.

Assuming that the number of training patterns is statistically sufficient to represent the input-output mapping $\mathfrak{M}$ for all the input space $\mathcal{X}$ [9], then the total average error measure $E$ can be approximated as:

\[ E(w, x) = \frac{1}{P} \sum_{p=1}^{P} e_p. \]

The training process can then be written as:

\[ \min_{w} \mathfrak{N}(w, x) = \min_{w} \frac{1}{P} \sum_{p=1}^{P} e_p, \quad \forall \ x_p \in \mathcal{I}, \ p = 1, ..., P, \]

such that $\mathfrak{N}(w, x) - \mathfrak{M}(x)$ $\leq e$, with $\mathcal{I}$ as the training data set and $e$ as the preset tolerance of the difference between $\mathfrak{N}(w, x)$ and $\mathfrak{M}(w, x)$ that satisfies Eq. (4).

In order to train a feedforward network, the backpropagation training algorithm is used to minimize the error measure in Eq. (7). The details of backpropagation training algorithm can be found in references such as [1]. In the backpropagation training algorithm, the change of weights $\Delta p w_{ji}$ in the network is:

\[ \Delta p w_{ji} = \eta \delta_{pj} o_{pi}, \]

where subscript $p$ denotes the $p$-th training pattern, $w_{ji}$ denotes the weight between the $j$-th neuron and $i$-th neuron from the layer below, $\eta$ is the learning rate, $\delta_{pj}$ is the backpropagating error, and $o_{pi}$ is the output of the $i$-th neuron for the $p$-th training pattern. It is a common practice to add a momentum term in Eq. (9) in an effort to avoid local minimum and speed up the training process:

\[ \Delta p w_{ji}^{itera+1} = \eta \delta_{pj} o_{pi} + \alpha \Delta p w_{ji}^{itera}, \]

where $\alpha$ is the momentum rate, and $itera$ is the training session number. For neurons in the output layer, $o_{pj} = y_{pj}$, and the $\delta_{pj}$ in Eq. (10) becomes:

\[ \delta_{pj} = -\frac{\partial E}{\partial y_{pj}} = -\frac{\partial E}{\partial y_{pj}} f_j, \]

where $f_j$ is the derivative of the activation function used in the $j$-th neuron. For neurons in the hidden layers,

\[ \delta_{pj} = \left( \sum_k \delta_{pk} w_{kj} \right) f_j, \]

where $k$ is the index of neuron in the layer above the $j$-th neuron.

3. Mean Square Error Measure

The mean-square error measure (or its modifications) is probably the most popular error measure used to train feedforward artificial neural networks. The mean-square error measure used in Eq. (7) is:

\[ e_p = \frac{1}{2m} \|y_p - y_p\|_2^2, \]

where $m$ is the dimension of the vectors $t$ and $y$, and $\frac{1}{2}$ is a scaling factor which makes it easier to take the derivative of the error measure, as used in many optimization problems. As indicated in Eq. (13), when output $y_p$ is same as the target value $t_p$, then the pattern error $e_p$ will be zero, which is the minimum of the error function. If the network output is not the same as the target, then the network learns to give the correct output with speed proportional to the descent direction $-\delta_{pj} \delta_{ypj} f_j$ as shown in Eq. (11). Actually, when different error functions are used, the only difference in the backpropagation training algorithm is in the output layer, for the term $\delta_{pj} \delta_{ypj}$ in Eq. (11). The form of $f_j$ and $\delta$ from hidden units remain the same because they are not explicit functions of the error function.
measure. The term $\frac{\partial e_p}{\partial y_{pj}}$ gives the rate of correction of the output error with respect to the network output, which for the mean-square error is:

$$\frac{\partial e_p}{\partial y_{pj}} = -(t_{pj} - y_{pj}).$$

(14)

Note that $f'$ is bounded between $(0, 1)$ when the popular sigmoidal activation function is used, thus the learning speed of the network is at most linearly proportional to the network error.

A classification problem with three classes is used to demonstrate the difference between using the mean-square error measure and the Bernoulli error measure. The discussion in this paper can be extended easily to multi-class classification problems. The classes are represented by 0, 0.5, or 1, respectively or expressed as $i \in \{0, 0.5, 1\}$. The error functions are shown in Fig. 1. As the error function indicates, the error $e$ is minimal and equals to zero, when the output of the network $y$ is same as the target value $i$. Otherwise, the error is a quadratic function between $y$ and $i$. The correction rates $\frac{\partial e_p}{\partial y_{pj}}$ for the mean-square error measure are plotted in Fig. 2. The shape and magnitudes of the derivative surface significantly affect the neural network training speed. As indicated in Fig. 2, $\frac{\partial e_p}{\partial y}$ is linearly related with the error distance between the network output $y$ and actual target value $i$. Assuming that $f'$ in Eq. (11) is constant, the learning speed is linear (of course $f'$ is not constant during training, but this assumption is reasonable for comparison purposes of the training speed with respect to different error functions to be discussed in later sections). When $y$ is equal to $i$ for all training patterns, then $\frac{\partial e_p}{\partial y} = 0$, and the network is completely trained.

Fig. 1. Error function plot of the three-class classification problem using the mean-square error measure.

Fig. 2. Plot of $\frac{\partial e_p}{\partial y}$ using the mean-square error measure.

4. Bernoulli Error Measure

The mean-square error measure may be a suitable measure for training artificial neural networks to solve specific problems, but it may not be as suitable as the Bernoulli error measure when classification problems are of concern. The Bernoulli error measure is defined as:

$$e_p = -\sum_j \left( t_{pj} \log(y_{pj}) + (1 - t_{pj}) \log(1 - y_{pj}) \right),$$

(15)

where $t_p = [t_{p1}, t_{p2}, ..., t_{pm}]^T = [t_1, t_2, t_3, ..., t_p]$ is the vector of desired target classes, and $y_p = [y_{p1}, y_{p2}, ..., y_{pm}]^T$ is the output of the network as mentioned in previous sections. As indicated in Eq. (15), even when the network output is same as the target value, the error $e_p$ is not necessarily zero. For example, when $t_{pj} = 0.5$ and $y_{pj} = 0.5$, the error measure $e_p$ is 0.3. However, when $y$ matches $t$, it does give the optimal point of the error function. Let's analyze $\frac{\partial e_p}{\partial y_{pj}}$ in the output layer in Eq. (11) to investigate the network learning speed by using the Bernoulli error measure:

$$\frac{\partial e_p}{\partial y_{pj}} = \frac{1 - t_{pj}}{1 - y_{pj}} \cdot \frac{t_{pj}}{y_{pj}}.$$

(16)

As indicated in Eq. (16), when $y_{pj}$ is same as $t_{pj}$, the quantity $\frac{\partial e_p}{\partial y_{pj}} = 0$, thus the network is completely trained. Otherwise, the network weights will be trained according to $(\frac{\partial e_p}{\partial y_{pj}}) f'$, where $\frac{\partial e_p}{\partial y_{pj}}$ is described in Eq. (16).

Using the same three-class classification problem described in Section 3, Fig. 3 shows the corresponding Bernoulli error function. For the three-class problem using the Bernoulli measure, $\frac{\partial e_p}{\partial y_{pj}}$ is plotted in Fig. 4. As indicated in Eq. (16), the learning speed of the network on the output layer is non-linearly related to the output error when the Bernoulli error measure is used, which gives a much faster convergence rate compared to the mean-square error measure when the output is far away from the target values. The next section will compare the learning rates between the Bernoulli error measure and the mean-square error measure.
5. Learning Speed Comparison between Bernoulli Error Measure and Mean Square Error Measure in Classification Problems

Due to the complexity of the feedforward neural network's learning dynamics, it is difficult to analyze the two different error measures in a rigorous closed form solution. But by making reasonable assumptions, the performance profile can be compared.

The speed of the network training is defined as the number of iterations it takes to meet the specified network termination criteria as stated in Eqs. (4) and (5). As the network learns, its weights change in an effort to meet the given criteria. Therefore, the change in weights in Eq. (9) govern the learning speed of the network. In Eq. (9), the error is backpropagated through $\delta_{pj}$, which is a function of the chosen training error measure. Let's assume that sigmoidal neurons are used in the network. Then the mean-square error measure gives $\delta_{pj}$ in the output layer as:

$$\delta_{pj}^{(ms)} = (y_{pj} - y_{p}) (1 - y_{p}) y_{pj}. \quad (17)$$

The superscript $(ms)$ in Eq. (17) signifies that $\delta$ is obtained from the mean-square error measure. On the other hand, when using the Bernoulli error measure for the sigmoidal neuron in the output layer, $\delta_{pj}$ results in:

$$\delta_{pj}^{(B)} = \frac{1 - y_{pj}}{1 - y_{pj}} \frac{1 - y_{p}}{y_{pj}} (1 - y_{p}) y_{pj} = (y_{pj} - y_{p}). \quad (18)$$

Again, the superscript $(B)$ in Eq. (18) indicates that $\delta$ is obtained from the Bernoulli error measure. The term $(1 - y_{pj}) y_{pj}$ is the only difference between Eqs. (17) and (18) or:

$$\frac{\delta_{pj}^{(ms)}}{\delta_{pj}^{(B)}} = (1 - y_{p}) y_{pj}. \quad (19)$$

Obviously, the learning speed from using the mean-square error measure and the Bernoulli error measure depends only on the output values of the neuron, $y_{pj}$. When the unipolar sigmoidal activation function is used, $y_{p} \in (0, 1)$ and $(1 - y_{pj}) y_{pj}$ is bounded between $(0, 0.25)$. The relationship between $\delta_{pj}^{(ms)}$ and $\delta_{pj}^{(B)}$ is graphically displayed in Fig. 5 as a function of $y_{pj}$. Eq. (19) and Fig. 5 indicate that $\delta^{(B)}$ is at least four times larger than $\delta^{(ms)}$, when $y_{pj} = 0.5$. In other words, the $\delta$ of a network using Bernoulli error measure is at least 4 times larger than the $\delta$ obtained from using mean-square error measure. As $y_{pj}$ deviates from 0.5, the relationship of $(1 - y_{pj}) y_{pj} $ makes the difference between $\delta_{pj}^{(ms)}$ and $\delta_{pj}^{(B)}$ even more significantly as shown in Fig. 5.

![Fig. 5. Relationship between $\delta_{pj}^{(B)}$ and $\delta_{pj}^{(ms)}$ with sigmoidal activation function used as a function $y_{pj}$.](image)

In this paper, a linearly separable AND logic classification problem (as shown in Table I) and a nonlinearly separable XOR logic classification problem (as shown in Table II) are used to illustrate the training speed and stability region differences between the mean-square error measure and the Bernoulli error measure.
Table I. AND problem.

<table>
<thead>
<tr>
<th>$p$</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$t$ = $\mathcal{N}(x_p)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table II. XOR problem.

<table>
<thead>
<tr>
<th>$p$</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$t$ = $\mathcal{N}(x_p)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Fig. 6 and Fig. 7 show typical training performances for the AND and XOR of a 2-input, 2-hidden, and 1-output node feedforward net structure, $\mathcal{N}_{2,2,1}$, using a learning rate $\eta$ of 0.1, momentum term $\alpha$ of 0.5, and initial weight $w_{ij} \in [-0.5, 0.5]$, and a stopping criteria of $\parallel w \parallel \leq 0.0001$. Two norm error measure of $\parallel e \parallel_2 = \parallel y - t \parallel_2$, which gives the physical Euclidean distance between the network output values and the target values in the output space, is monitored when this network is trained by (1) mean-square error measure and (2) Bernoulli error measure.

Fig. 6. Typical AND training error performance using different error measures.

Fig. 7. Typical XOR training error performance using different error measures.

We also compare the stability region of the training using the mean-square error measure and the Bernoulli error measure, given different learning rates and momentum rates as expressed in Eq. (10). The *stable region* is defined as the training will converge to a solution in a significantly long training time for the specific problem, say 100,000 epochs for the AND logic training. *Unstable region* is thus defined as the training will not converge to a solution within a significantly long training time for the problem. To avoid local minimum problems or outlier events occurring during training that may affect our analysis, the training experiments are repeated several times and the mean of the results is used to represent the stability region.

The stability regions to train a feedforward net to learn the AND logic using the mean-square error measure and the Bernoulli error measure are shown in Figs. 8a and 8b, respectively. The stability region for the mean-square error measure is significantly larger than the one for the Bernoulli error measure. A quick estimate on the stability region, $A$, gives: $A^{(ms)} = 6.5$ per unit, $A^{(B)} = 1.5$ per unit. The stability regions to train a feedforward net to learn the XOR logic using the mean-square error measure and the Bernoulli error measure are shown in Figs. 9a and 9b, respectively. A quick estimate on the stability region gives: $A^{(ms)} = 5.8$ per unit, $A^{(B)} = 1.3$ per unit.

As discussed previously, the training of classification problems using the Bernoulli error measure is much faster than that of the mean-square error measure. Although the analysis in Eq. (19) does not fully describe the actual training process (which is too complicated for analysis in a closed form solution), it does provide a reasonable good comparison between the learning speeds of the mean-square error measure and Bernoulli error measure as shown in the simulation results for AND and XOR illustrated in this paper.

Fig. 8a. Stability region for AND classification for $\mathcal{N}_{2,2,1}$ using the mean-square error measure.

Fig. 8b. Stability region for AND classification for $\mathcal{N}_{2,2,1}$ using the Bernoulli error measure.
Fig. 9a. Stability region for XOR classification for $\mathcal{N}_{2-2-1}$ using mean-square error measure.

Fig. 9b. Stability region for XOR classification for $\mathcal{N}_{2-2-1}$ using Bernoulli error measure.

On the other hand, the stability region using the Bernoulli error measure is significantly smaller than the stability region using the mean-square error measure. The results are consistent with many engineering problems in that there are always trade-offs between learning speed and stability. It will be challenging to search for a relationship between the stability regions and the learning speeds, if any.

6. Conclusions

This paper has described and illustrated the use of the Bernoulli error measure to train feedforward artificial neural networks for classification problems. The Bernoulli error measure's performance is compared to the most popular mean-square error measure. Successful training using the Bernoulli error measure does not require that the error function goes to zero. Rather, the solution must be at the point where the first derivative of the training is zero, as analyzed in the paper. Significant increase in training speed has been obtained by using the Bernoulli error measure instead of the mean-square error measure. Even though the stability region for successful training for the Bernoulli error measure is much smaller than the one for the mean-square error measure, the stability region for the Bernoulli error measure does cover the training parameters we normally use for network training. With appropriate choices of training parameters, the Bernoulli error measure is certainly a potentially appropriate choice to train a network to learn classification problems. More research including robustness, noise effects, etc. of the Bernoulli error measure will be carried out and reported in the future.

Acknowledgment

The authors of this paper would like to acknowledge the support of the (1) National Science Foundation, for Grant No. ECS-8922727, (2) Electric Power Research Institute - Exploratory and Applied Research Division, for Research Contract RP8004-24.

Reference