Depth Learning Standard Deviation Loss Function

Chunlin Wang¹,a, Jianyong Sun¹,b, Wanjin Xu¹,c, Xiaolin Chen¹*

School of Information Science & Technology, Chuxiong Normal University, P.R.China

*Corresponding author e-mail: chenxl@ctxc.edu.cn
awcl@ctxc.edu.cn, b sunny@ctxc.edu.cn, cxwj cx@ctxc.edu.cn,

Abstract. Deep learning is a branch of the field of machine learning that outperforms humans in image and natural language processing. This mainly benefits from powerful computer processing capability and massive datasets. In the training model, the loss function needs to be used to evaluate each training result. The quality of the loss function directly affects the correctness and validity of the model. In this paper, four kinds of common loss functions in deep learning are studied and our own loss function is proposed; Then the MNIST dataset is adopted to classify and train them; Finally, in the course of training, the change and the classification correct rate of the loss function value and model parameters are observed. Firstly, the different loss functions are experimented and their performance and application fields are analyzed, and secondly, our loss function is defined. After 30,000 times of iterative training, the test set data is used to test the loss function proposed, the correct rate reaches 98.53%, and achieves better accuracy in fewer training iterations, saving training times and resources.

1. Introduction

In recent years, artificial intelligence has achieved more brilliant results in many fields, such as image processing and natural language processing, which benefit from the strong computer processing power using neural networks to process and count large amounts of data sets. Deep Learning promotes the wide application of neural networks in many fields. In the deep learning model training, the loss function is used to evaluate the results of each training, the model parameters are adjusted according to the training results of the previous round, and then the next iterative training is carried out until the loss function converges.

Deep learning is a new field in machine learning research, the purpose of which is to simulate the neural networks in charge of learning in the human brain, to allow computers to learn from experience, and to understand the world according to hierarchical conceptual systems, while each concept is defined by the relationship with some relatively simple concepts. Allowing a computer to acquire knowledge from experience can prevent humans from formalizing all the knowledge it needs for a computer. The concept of hierarchy allows computers to build relatively simple concepts to learn more complex concepts and knowledge. If a graph of how these simple concepts are built on each other can be obtained, another deeper picture can also be gotten. It represents a property category or feature by combining lower-level features to form a more abstract upper layer to discover the distributed feature representation of the data. At the same time, it is also a kind of combination of multi-layer nonlinear transformation modeling algorithm for high complexity data. The method to realize multilayer
nonlinear transformation is the deep neural network model, as shown in Figure 1, which contains a fully connected network of an input layer, a hidden layer and an output layer, and the Input on the left side of the graph is a eigenvector X as inputs to the neural network, and on the right is the output layer Y', the neural network node between the input and output layers is the hidden layer. The more the hidden layers, the deeper the neural network. The collection of weights for each node of the hidden layer is called vector W. The Formula 1 is used to calculate the eigenvector x of the neural network input layer and the weight W of the hidden layer to obtain Y', the difference between the Y' and the expected value Y by calculating the cross-entropy Formula 3 of the loss function, and then the gradient of each parameter is calculated for the reverse propagation optimization algorithm. According to the learning rate \( \eta \), using the gradient descent algorithm to activate the function Formula 4 to obtain the next round weight \( W^{n+1} \), through N times of iterations to make the loss function value smaller, and finally, the weight \( W^n \) is obtained, that is, the model obtained after the neural network training.

\[
\begin{bmatrix} y_1 & y_2 & \cdots \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & \cdots \end{bmatrix} \begin{bmatrix} W_1' \\ W_2' \\ \vdots \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \cdots \end{bmatrix} \tag{1}
\]

\[
Y' = f(\sum_i X_i W_i + b) \tag{2}
\]

\[
\text{Loss}(Y, Y') = \sum_{i=1}^{n} f(y'_i, y_i) \tag{3}
\]

\[
W^{n+1} = W^n - \eta \frac{\partial}{\partial W^n} J(W^n) \tag{4}
\]

At present, the main application areas of deep learning are image recognition, Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) processed by natural language.

**Figure 1.** Full Connection Neural Network Model  
**Figure 2.** Convolutional Neural Network

CNN is a neural network designed to process data with similar network structures. Later, it is improved to evolve image recognition and detection algorithms such as ImageNET, R-CNN, Fast R-CNN, and Faster R-CNN [2, 3, 4]. The difference between it and the fully connected network lies in the connection between the adjacent layers of the neural network, in order to deal with the more complex problems, the partial connection is adopted between adjacent layers, which can not only reduce the dimension and computation, but also avoid the problem of slow calculation speed and over-fitting due to too many parameters. It is generally composed of five parts, which is shown in Figure 2[5]: 1. Input layer: The eigenvalue is entered as a neural network, such as image matrix; 2. Convolutional Layer, it carries on the deep analysis to the previous layer structure, thus gets the more abstract characteristic vector, after the layer processing, the structure becomes deeper; 3. The Pooling layer, after which it is processed to obtain a smaller image matrix, but the depth remains the same. 4. Full connection layer, after processing by convolutional layer and pooling layer, the full connection layer is used to complete the classification task, the number of nodes in the layer is determined by the number of categories; 5. Softmax layer, which compresses the value range of the fully connected layer to (0,1) and makes it conform to the probability distribution.

RNN originates from the Hopfield network proposed by Saratha Sathasivam in 1982, which is replaced by traditional machine learning algorithms in the early days. However, Because the characteristic that traditional machine learning algorithm relies on manual extraction, the problem of image recognition, speech recognition and natural language processing based on traditional machine learning has the bottleneck of feature extraction. Due to there are too many parameters, the fully connected neural network model is not good at processing data time series information [6]. Then more effective RNN structure is constantly proposed, in the mining data, the deep expression ability of time
series information and language information are fully utilized, and has achieved a breakthrough in speech recognition, language model, machine translation and other natural language processing. RNN is mainly used to process and predict sequence data. The structure of which is shown in the figure, which can be considered as a feedforward neural network with N intermediate layers after the expansion of the RNN. This type of feedforward neural network does not have circular links, so it can be trained directly using a back propagation algorithm without the need for special optimizations or operations. Back propagation is one of the common methods to train RNN. For sequence data, the RNN can be entered into the eigenvalue in turn on different time series, the state S of the current timing is calculated, and the S is passed to the next sequence to continue to calculate forward until the output value is calculated. Then the gap between the output value and the expectation is calculated by the loss function, subsequently, the matrix parameters of the RNN are updated in reverse, and the optimal results are achieved through numerous training sessions. With the use of machine learning in natural language processing, in the course of RNN training, the length of the sequence may lead to the problems, such as the optimization of gradient dissipation and gradient explosion [7]. At the same time, in order to better solve the long-term dependence of RNN, the LSTM [8] is proposed in 1997 to solve the problem. In the treatment of complex problems, LSTM has a better performance than the standard RNN. In recent years of research, structures such as Bidirectional RNN, Deep RNN have been proposed.

2. Loss function

In image and time series data, different samples are classified and predicted by the above neural network model. In model training, how to judge whether the model processing results are consistent with the results defined in advance? In deep learning and neural networks, the loss function is introduced to solve this problem, and the loss function is used to judge the quality of the model. A good loss function will improve the reliability and correctness of the model and be used to calculate the deviation between the model results and the correct results. According to the deep learning model and the problems to be solved, the loss functions can be divided into two categories: Classification problem loss function and regression problem loss function. This paper mainly discusses the classification loss function. At present, the loss functions commonly used in the classification model are cross-entropy and mean squared error (MSE).

In the classification problem, the most common method to solve the multi-classification problem through neural network is to set n output nodes, where n is the number of categories. For each sample, the neural network outputs an n-dimensional vector as a result. Each dimension in the vector corresponds to a category. Ideally, if a sample belongs to the category \( Y_k \), then the output value of the K dimension of the output node after the sample is processed by the pattern should be 1, while the outputs of the other nodes are 0. Taking the classical MNIST recognition number as an example, it has a total of n=10 numbers as output nodes, recording the probability that a number belongs to different categories, respectively. Ideally, the closer the output of the model to [0,0,0,1,0,0,0,0,0,0], the better the effect. However, the actual output value of the model is [0.1, 0.02, 0.2, 0.8, 0.11,0.1,0.01,0.02,0.02]; through the analysis of model output results, it can be seen that the probability that the sample belongs to the number "3" is the largest, then the model judges it as "3". Cross entropy depicts the distance between two probability distributions, which is originally used to estimate the average encoding length. Given the two-phase probability distribution \( p, q \), the cross entropy of \( Y' \) is represented by \( Q \) as:

\[
H(Y, Y') = -\sum_x Y(x) \log Y'(x) \tag{5}
\]

From the formula of cross entropy, it can be seen that the cross entropy function is not symmetrical ( \( H(Y, Y') \) ) \( \neq (H(Y', Y) \) ), it depicts the difficulty of expressing the probability distribution \( Y' \) through the probability distribution \( Y' \). Because the correct answer is the result wanted, so when cross entropy is used as a loss function for neural networks, the \( Y \) code is the correct answer, and \( Y' \) represents the predicted value. The smaller the cross entropy value, the closer the two probability distribution, the better the training effect, and the easier the neural network model converges.
However, the output of the neural network does not necessarily conform to the probability distribution, such as the above example of mathematical classification. Its probability distribution depicts the probability of different events occurring. In the case of a limited total number of events, the probability distribution function $p(x=x)$ satisfies Formula 6.

$$\forall x \in [0,1] \sum_x p(x=x) = 1 \quad (6)$$

That is, the probability of any event is between 0 and 1, and there is always an event (probability and 1). The "sample belongs to a certain category" in the classification problem is regarded as a probabilistic event, then the correct answer to the training data conforms to a probability distribution. Since the probability of the event "a sample is an incorrect category" is 0, and the probability of "a sample belongs to the correct category" is 1, in a neural network, the Softmax regression is used to solve such problem at the output layer. Assuming that the original neural network output is $y_1, y_2, \ldots, y_n$, then after Softmax regression processing, the output is:

$$\text{softmax}(y)_i = \frac{e^{y_i}}{\sum_{j=1}^{n} e^{y_j}} \quad (7)$$

As can be seen from Formula 7, the output of the original neural network is used as a confidence degree to generate a new output to meet all the requirements of the probability distribution. This new output can be understood as: After neural network derivation, how large the difference between the probabilities of one example for different categories. The numerical classification examples in this section need to use formula 7 to satisfy the conceptual distribution.

3. Our loss function

In deep learning, the loss function is almost used by all training models to evaluate each training result, and then the back propagation optimization algorithm is adopted to dynamically adjust the parameters in the model according to the evaluation. In the training process, the parameters are constantly adjusted, so that the loss function gradually converge to 0. The following is a discussion of four loss functions.

3.1. Sigmoid Cross Entropy

The implementation of Sigmoid Cross Entropy is the same as that of the previous crossover entropy algorithm, and it is also the earliest cross entropy algorithm implemented by TensorFlow. It is calculated by Formula 8 for the vector $\text{logits}$ output of the model $\hat{y}$, which is the classification result, while $y$ is the correct classification result. Then, Formula 9 is used to calculate their loss value. It is often used in multi-objective problems that are independent but non-mutually exclusive between classifications, such as judging whether a picture contains a variety of animals.

$$y'_{ij} = \text{sigmoid}(\text{logits}_{ij}) = \frac{1}{1+e^{-\text{logits}_{ij}}} \quad (8)$$

$$\text{loss}_{ij} = y_{ij} \ln(y'_{ij}) + (1 - y_{ij}) \ln(1 - y'_{ij}) \quad (9)$$

3.2 Softmax Cross Entropy

For the output vector logits, Formula 10 is used to calculate, firstly, the purpose of which is to compress the Logits domain value in (0,1), so that the result does not overflow, and the probability distribution is consistent, that is, the sum of all probabilities is 1. Then the loss value is calculated by Formula 11. It applies to models that are independent and exclusive to each category, such as the age group, any person belongs to only one of these stages and cannot belong to multiple stages, simultaneously.

$$y'_{ij} = \text{softmax}(\text{logits}_{ij}) = \frac{e^{\text{logits}_{ij}}}{\sum_{j=0}^{\text{numclasses}-1} e^{\text{logits}_{ij}}} \quad (10)$$

$$\text{loss}_i = -\sum_{j=0}^{\text{numclasses}-1} y_{ij} \ln(y'_{ij}) \quad (11)$$

Softmax cross entropy is not the same as sigmoid_cross_entropy, although all vector dimensions are the same, the result of requiring classification is mutually exclusive, ensuring that only one field has a value. If the multi-objective problem is involved, after Softmax, it is not possible to get multiple cases where the sum is 1. And if the number of label is more than 1, the cross entropy can not be
calculated, so this function is only suitable for the single target of two classification or multi-classification problem.

3.3 Sparse Softmax Cross Entropy

The Sparse Softmax Cross Entropy is a simplified version of Softmax Cross Entropy, which also applies to situations where each category is independent of each other and is excluded. However, it calculates the multidimensional features of onehot encoding within the loss function, so it is more versatile.

3.4 Weighted Cross Entropy

The weighted_sigmoid_cross_entropy_with_logits is an expanded version of Sigmoid_cross_entropy_with_logits, the input parameters and implementations of the former are the same as those of the latter, except that they support the weights pos_weight parameters, in order to increase or decrease the loss of the positive sample when calculating cross entropy. The implementation principle is very simple, in the traditional cross-entropy algorithm based on sigmoid, the value calculated by the positive sample multiplied by a certain coefficient interface, through the weight pos_weight to control the change of model parameters, and reduce the loss function left and right fluctuation on the correct rate of the whole model. The algorithm substituting Formula 8 into Formula 9:

\[
\text{loss}_{ij} = -[\text{pos_weight} \times y_{ij} \ln(y'_{ij}) + (1 - y_{ij})\ln(1 - y'_{ij})]
\]

12

3.5 The definition of our loss function

In our loss function defines, first, the formula 11 is obtained by substituting Formula 8 into Formula 10. Firstly, the predictive value vector \(Y'\) domain value is mapped between (0,1), and the difference square value of the calculated value of the model \(Y'\) and the ideal value \(Y\) is obtained. Then, the average of the internal variance of batch is calculated. The main function of the sigmoid function is to optimize the calculation, so that the results do not overflow, which is suitable for each category of independent but non-exclusive models. \(Y'\) is a vector value, the dimension is equal to the number of categories N, but the loss of each sample in a batch, so the average value of the loss function is generally sought in batch.

\[
\text{loss} = \frac{\sum_{i,j}(y_{ij} - y'_{ij})^2}{n}
\]

(13)

\[
\text{loss} = \frac{\sum_{i,j}(y_{ij} - \frac{1}{1 + e^{-\text{logits}_{ij}}})^2}{n}
\]

(14)

In formulas 8 and 9, the input parameters are \(\text{logits}_{ij}\) and \(Y\), \(\text{logits}_{ij}\) is the result of \(x\times w\) matrix multiplication in the neural network model, while the dimension of \(Y'\) is the same as \(\text{logits}_{ij}\), and \(Y\) is the correct label value. For example, a digital image of batch=100 28*28 pixels is trained once in the MNIST handwritten mathematical recognition model, the input vector \(X[\text{batch,28,28}]\) is the input value of the three-dimensional vector as the model, and the two-dimensional vector is \(X[\text{batch,784}]\) through the dimension conversion. Then with the model weight parameter \(w[784,700]\), the matrix multiplication operation is carried out, 700 is the two-dimensional vector of the hidden layer node, the \(\text{logits}\) \([100,10]\) is obtained, then the \(Y'\) \([100,10]\) is obtained by using Formula 8, and the difference value of the loss function is calculated by Formula 13 and 14. According to the difference square value and learning rate of the loss function, the parameter \(W\) in the model is dynamically adjusted by the back propagation optimization algorithm, the \(N=10\) is the number of categories, the categories are independent, and \(y[100,10]\) represents the correct category of the 100 images.

4. Experiment

The effect of loss function on recognition accuracy is tested by MNIST [12] handwritten mathematical recognition test in the section. The MNIST dataset has 60000 images as training data and 10000 images as test data. Each picture represents a math in 0-9 and is a single-classification problem. The input layer node is a picture with a pixel of 784=28*28; the first layer is a hidden layer with 700 nodes; and the output layer has 10 nodes, corresponding to a digit in 0-9, respectively. In the
experiment, the learning rate is 0.001 and the number of iterative training is 30,000. The model is optimized using the regularization and exponential Moving Average to improve the accuracy of recognition. In the experiment, the same models and datasets are used to test the different manifestations of different loss functions. In the following, the change of model parameter W will be observed from loss value and recognition accuracy.

![Figure 3. Change in loss function value](image)

### 4.1 Change in the value of the loss function

In **Figure 3**, the loss value of different loss functions decreases gradually with the increase of the number of iterative training, and the loss function slowly converges to 0, ours is very small, which is the smallest of all loss functions, and the smoothness decreases gradually during the iterative process. The remaining four loss function values fluctuate when the number of training times reaches 2,500 times, indicating that the gradient descent algorithm does not guarantee that the optimized function is the global optimal solution, and then the fixed learning rate used will also fluctuate. The fluctuations of Sigmoid Cross entropy and Softmax Cross Entropy are more pronounced, while the Sparse and Weighted are relatively smaller.

![Figure 4. Training process of model weight W](image)

![Figure 5. Training process of offset value B](image)

### 4.2 Model parameter comparison

In the training process of the model, in order to better observe the model weight W and offset value B matrix. First, the `Tf.summary.histogram()` function is used to compress the matrix into a histogram data structure consisting of widths and quantities, and the value range of the mean of its variance is compressed to (0,1). In the course of model training, the changes are observed, as shown in Figure 4 and Figure 5.

![Figure 6. The correct rate of five loss functions tested by using test sets](image)

### 4.3 Comparison of classification correctness

In the course of training, the Test dataset is constantly used for testing the trained model to detect the classification effect of the model. As shown in Fig6, the effects of random parameters at the beginning of training are significantly different, but as the number of training increases, the gap between them is getting smaller. After 30,000 times of iterative training, the classification accuracy of the loss function of ours reaches 98.53%, which is higher than the remaining four kinds of loss
functions. Weighted and sigmoid are better classified than Softmax and sparse, reaching 98.52%. When the number of iterative training is between 1K and 10K, ours is the most effective of all loss functions.

5. Conclusion

In this paper, the characteristics and uses of four kinds of loss functions are analyzed. When using and defining loss functions, the multi-objective and multi-classification scenarios should be understood, and choose an implementation based on sigmoid or Softmax depending on whether the application requirements are independent or mutually exclusive. We also defined our own loss function, after 30,000 experimental training, ours is achieves the best effect of all loss functions, the detection accuracy rate reaches 98.52%, and in fewer training iterations to achieve a better correct rate, saving training times and resources.

Acknowledgment

This work is supported in part by Yunnan Provincial Department of Education Project and the Science and Technology Program of Yunnan Province, China under grant No.2018JS454 and 2017FH001-124.

References

[1] LeCun Y, Bengio Y, Hinton G. Deep learning[J]. nature, 2015, 521(7553): 436.
[2] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]//Advances in neural information processing systems. 2012: 1097-1105.
[3] Ren S, He K, Girshick R, et al. Faster r-cnn: Towards real-time object detection with region proposal networks[C]//Advances in neural information processing systems. 2015: 91-99.
[4] Girshick R. Fast r-cnn[C]//Proceedings of the IEEE international conference on computer vision. 2015: 1440-1448.
[5] LeCun Y. LeNet-5, convolutional neural networks[J]. URL: http://yann. lecun. com/exdb/lenet, 2015: 20.
[6] Sathasivam S, Abdullah W A T W. Logic learning in Hopfield networks[J]. arXiv preprint arXiv:0804.4075, 2008.
[7] Pascanu R, Mikolov T, Bengio Y. On the difficulty of training recurrent neural networks[C]//International Conference on Machine Learning. 2013: 1310-1318.
[8] Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.
[9] http://colah.github.io/posts/2015-08-Understanding-LSTMs/
[10] https://blog.csdn.net/QW_sunny/article/details/72885403